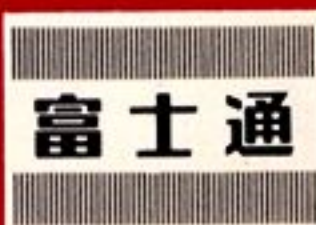
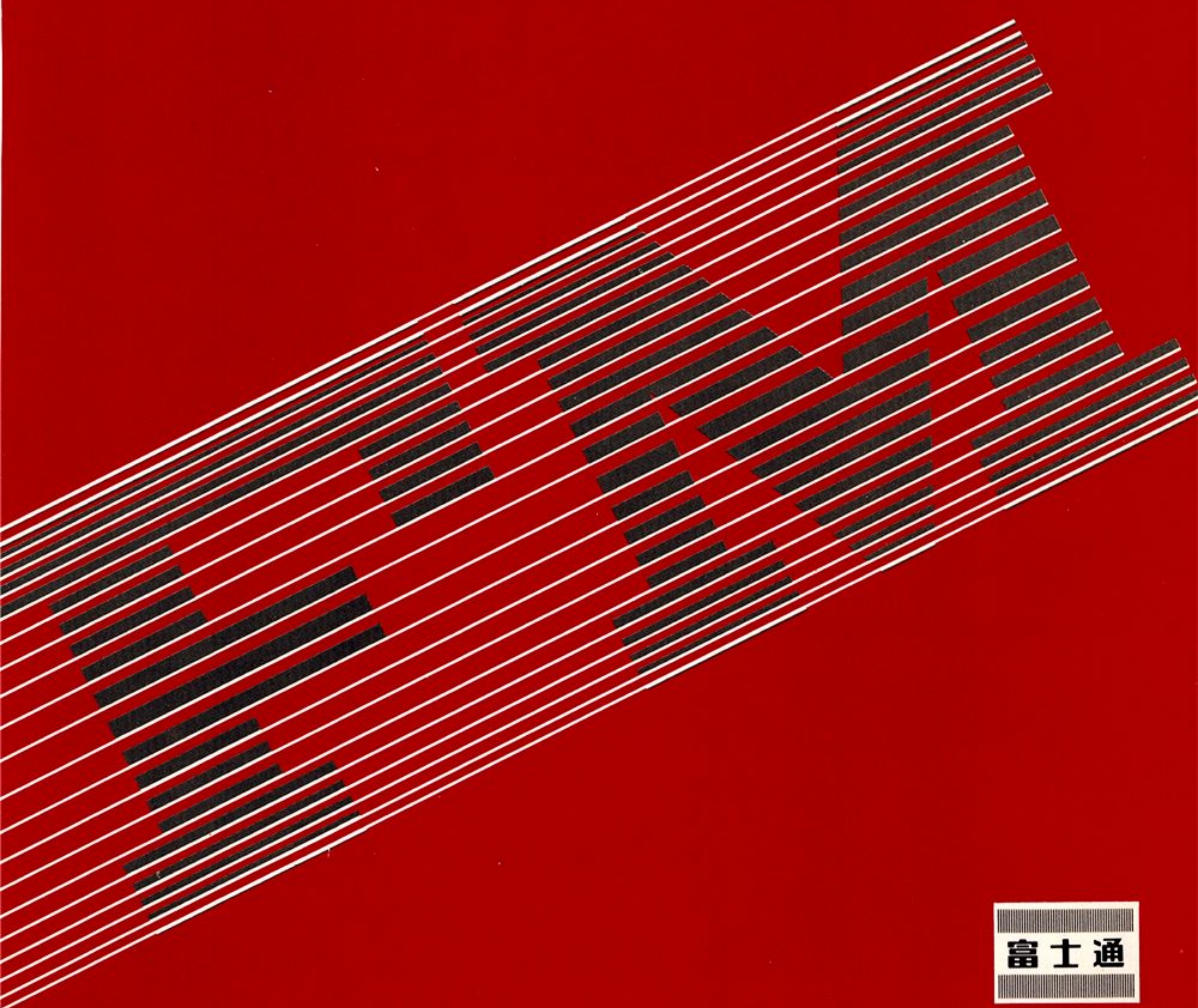
80EI-000010-2

パーソナルコンピュータ

FMシリーズ

ユーザーズマニュアル
F-BASIC入門

パーソナルコンピュータ

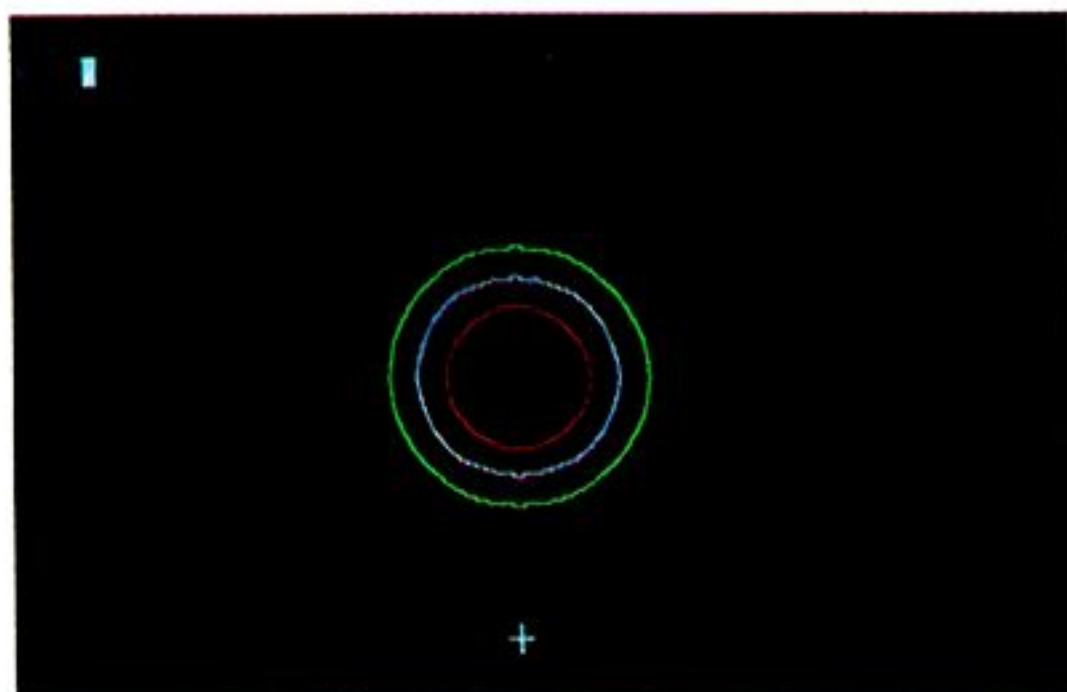
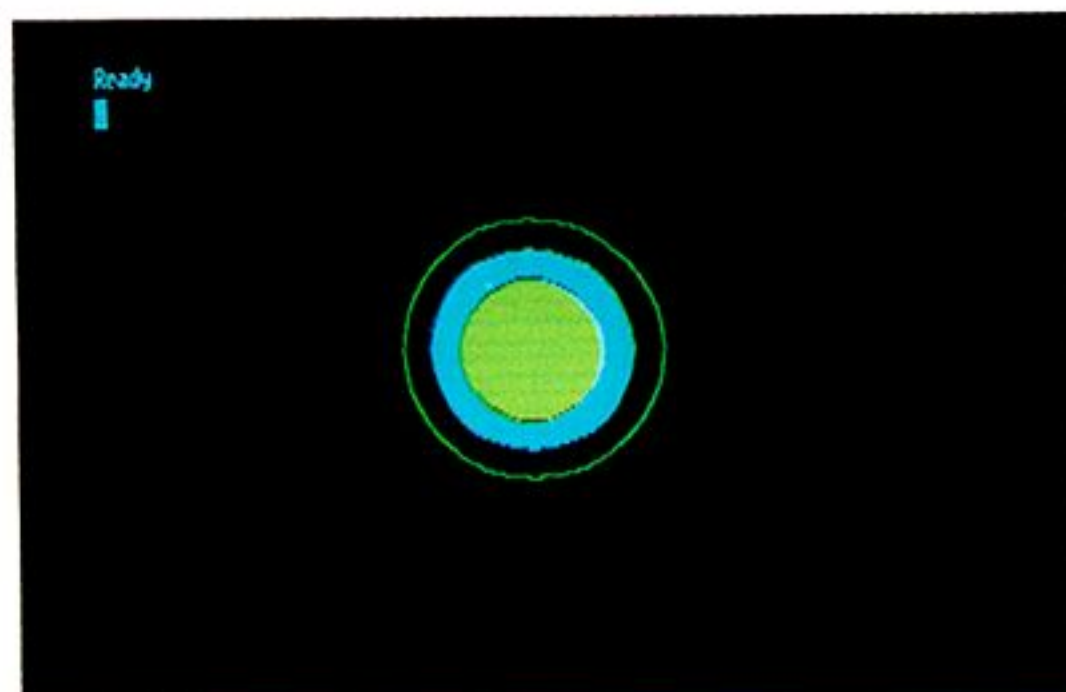
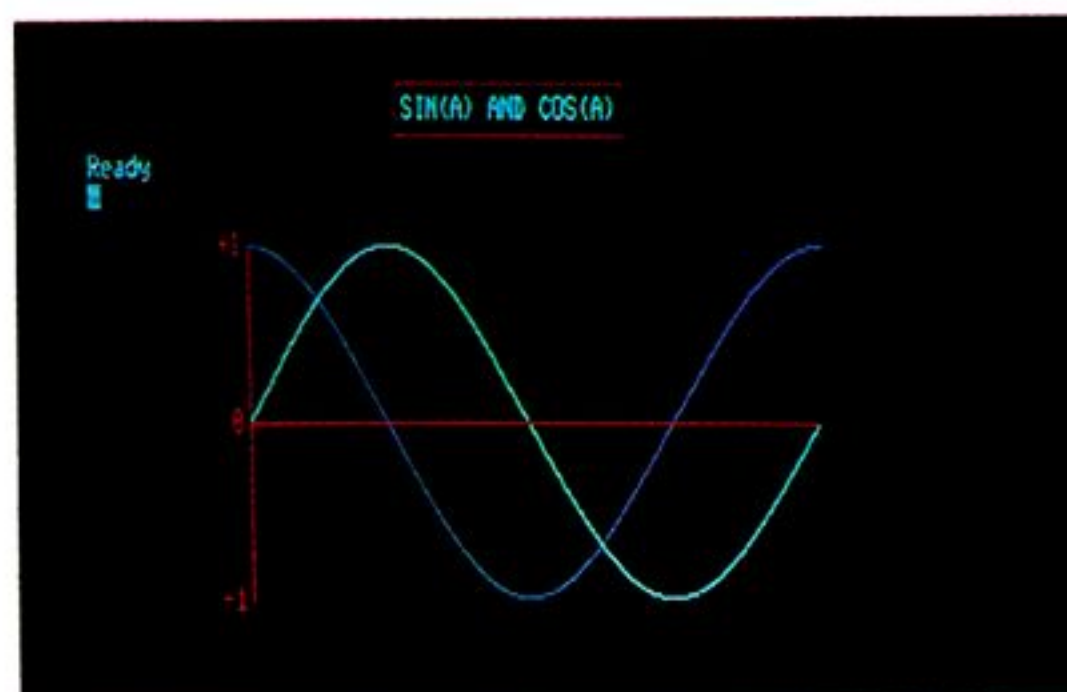
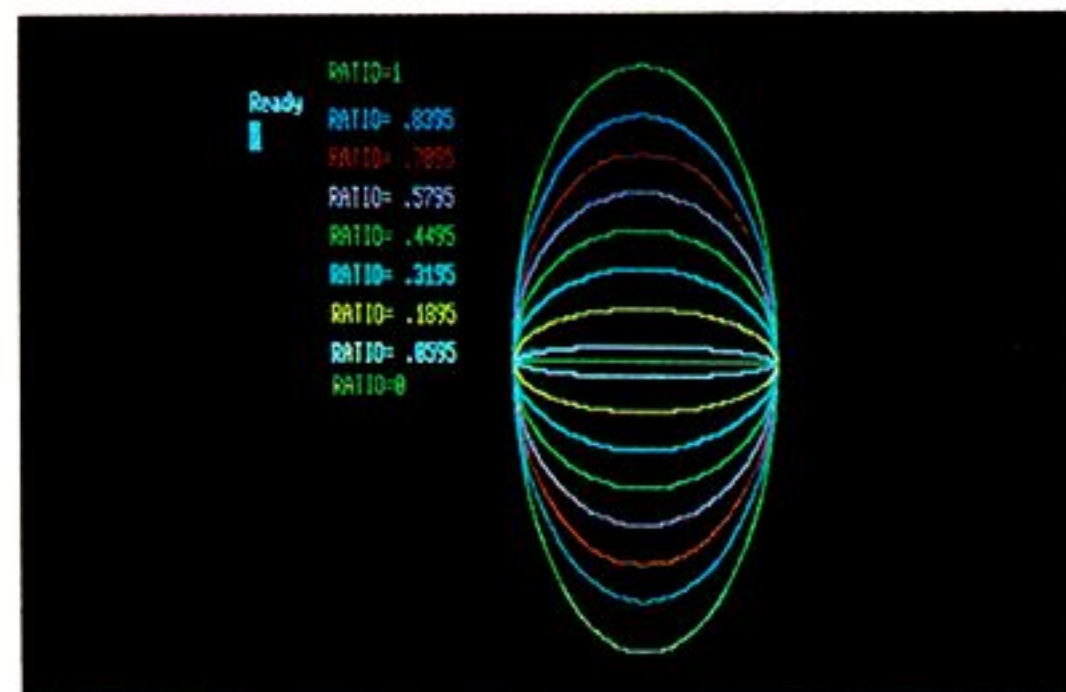
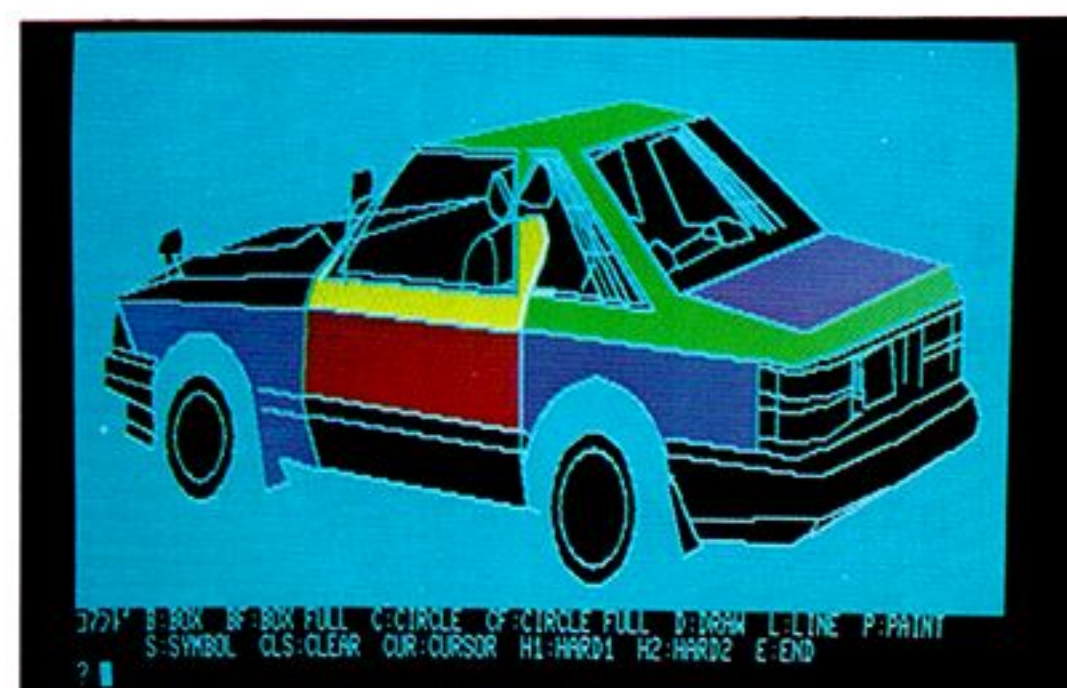
FMシリーズ

ユーザーズマニュアル F-BASIC入門

富士通株式会社

FM-7
本体外観

FM-8
本体外観

FM-11
本体と
キーボード
外観

GCURSOR，CIRCLE の例
（下方にカーソルの＋印が見える）

GCURSOR，PAINT の例
（色を塗りつぶした例）

SYMBOL　sin，cos 曲線の例

CIRCLE の例

いろいろな図形の操作
（たとえば，PAINT を使って車のボディーに好きな色を着色する）

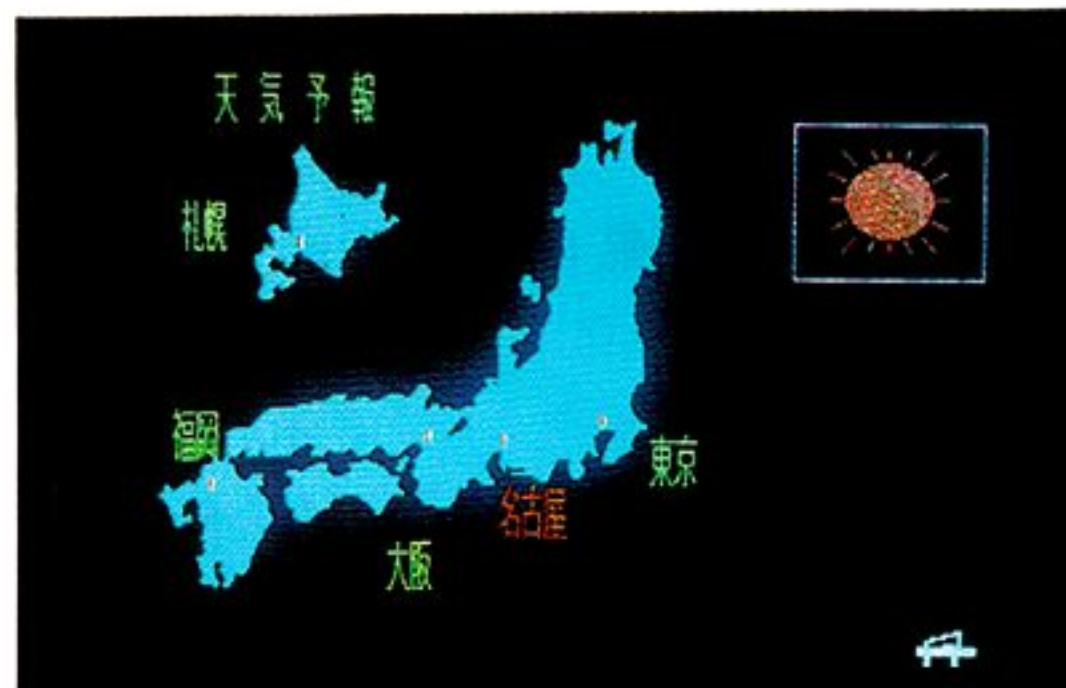
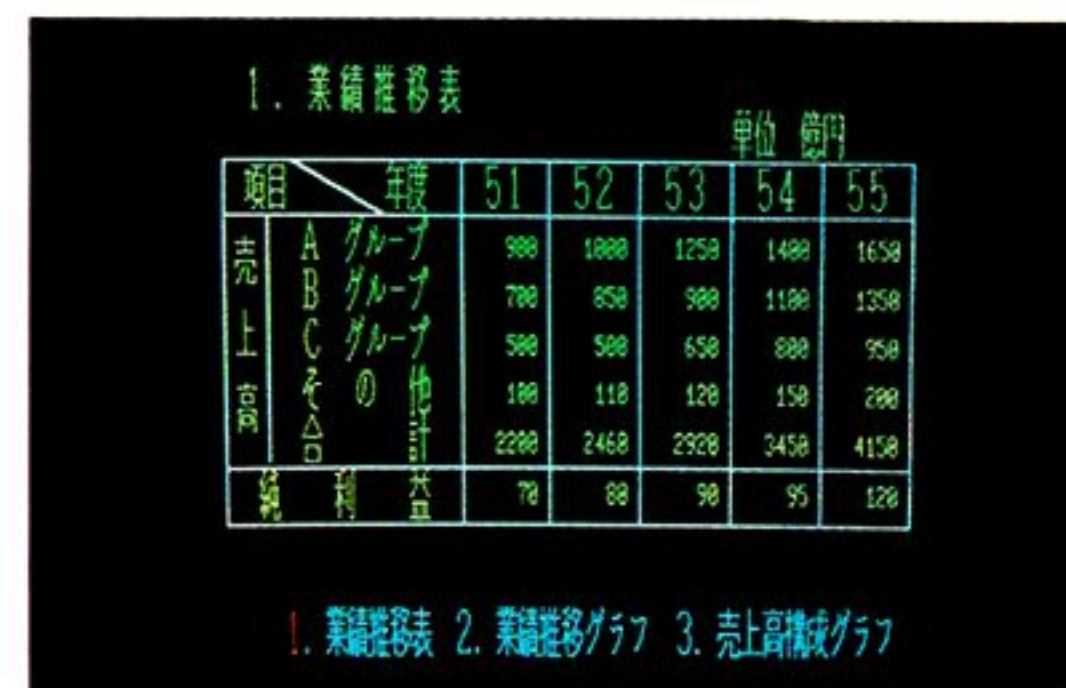
全国の天気予報の例

売上統計グラフの例

1．業績推移表

単位 億円

項目＼年度		51	52	53	54	55
売上高	Aグループ	900	1000	1250	1400	1650
	Bグループ	700	850	900	1100	1350
	Cグループ	500	500	650	800	950
	その他	100	110	120	150	200
	合計	2200	2460	2920	3450	4150
純利益		70	80	90	95	120

1. 業績推移表 2. 業績推移グラフ 3. 売上高構成グラフ

業績推移表の例

お　願　い

1．本書を始めとする各種マニュアルについてのお問合わせは，お買上げの販売店，および「富士通マイコンスカイラブ」へお願いいいたします．

2．本体および各種オプション品，周辺装置の取扱いについては，各種取扱説明書を充分にお読みのうえ，使用して下さい．

3．新製品ニュース，ソフトウェアについてのお知らせは，各種マイコン雑誌への広告および，保証書，アンケート用紙をご送付くださった方へのダイレクトメール等により行います．

富士通マイコンスカイラブ

虎ノ門：〒106　東京都港区虎ノ門２－３－13　第18森ビル内
TEL（03）591－1091，2561
月～金（祝日を除く）　９時30分～17時

秋葉原：〒101　東京都千代田区外神田１－15－16　秋葉原ラジオ会館６Ｆ
TEL（03）251－1448
年中無休　10時～19時

札　幌：〒060　札幌市中央区南一条西３丁目　丸井今井一条本館４Ｆ
TEL（011）241－4185
月～金（水曜定休日）　10時～18時
土・日　10時～18時30分

仙　台：〒980　仙台市国分町１－７－18　明治生命仙台国分町ビル１Ｆ
TEL（0222）66－8711

名古屋：〒460　名古屋市中区栄１－５－22　富士通ＯＡショールーム内
TEL（052）221－6016
月～土（祝日を除く）　10時～18時

大　阪：〒530　大阪市北区梅田１－２－２　大阪駅前第２ビル１Ｆ
TEL（06）344－7628
年中無休　10時～19時

広　島：〒733　広島市中区立町４－２　大橋ビル２Ｆ，３Ｆ
TEL（082）247－3949
年中無休　10時～19時

はじめに

パーソナルコンピュータでの初心者用プログラミング言語であるBASIC(Beginner's All purpose Symbolic Instruction Code)言語は，コンピュータになじみのなかった方にでも容易にプログラミングが可能になるとされる言語です．

しかしながら，難解な専門用語が次々と出て来る解説書では，一念奮起して理解しようとした意志が途中で挫折してしまいます．

そこで，本「ユーザーズマニュアル　F-BASIC入門」は，科学技術分野でのマンガ家として活躍されている緒方健二氏に，全体の構成，内容について全面的に依頼しました．また，理解を助けるためのイラストを水谷たけ子氏に描いていただきました．

本書は，全くの初心者を対象とした富士通パーソナルコンピュータFMシリーズ用F-BASIC入門書であり，F-BASICを理解された段階では，別冊の「F-BASIC文法書」，及び「ポケットブック」を御利用くださると，プログラムの作成，エラーの対処等に有効と考えます．

昭和58年8月

目　次

イラスト　水谷たけ子

FMシリーズ

LIST
10 REM F3.10-2
20 CLS
30 CIRCLE (300,100),50,2
40 CIRCLE (300,100),70,3
50 CIRCLE (300,100),90,4
60 GCURSOR (300,100),(X1%,Y1%),7
70 GCURSOR (300,100),(X2%,Y2%),1
80 PAINT (X1%,Y1%),6,2,3,4
90 PAINT (X2%,Y2%),5,2,3,4,6
Ready

1 パーソナルコンピュータ FMシリーズ

富士通パーソナルコンピュータFMシリーズには，FM－7，FM－8，FM－11と利用の目的に応じて各種の機種が揃っています．いずれも最新のエレクトロニクス技術が駆使された，高性能でコスト・パフォーマンスの優れたパーソナルコンピュータ（以後パソコンと略することにします）です．そして，あなたの仕事や勉強の手伝いに，あるいはコンピュータゲームなどの楽しみの相手としても，扱い易い機械になっています．

FMシリーズは，超大形コンピュータから中，小形，ミニコン等々さまざまのコンピュータや各種の通信情報機器を産み出してきた富士通が，最新のエレクトロニクス技術の粋を集めて作り上げたパソコンです．

パソコンとはいえ，大きなコンピュータ装置にも負けないほど高性能の機械ですから，数多くの機能を持っており，その扱い方については一通りの知識を持っていただかなくてはなりません．

でも，その仕事はこの本におまかせください．基本的な重要な使い方から順に述べてありますから，何の予備知識もなく，パソコンに接するのが初めての方でも順を追って読んでいけば，次第にその扱い方やその活用の仕方が理解できるようになっています．この本を読み，実際にFMシリーズのパソコンを扱ってみれば，これらの機械は，あなたの意のままに動かせるようになるに違いありません．これを元に，あなた自身が創造したパソコン活用法を楽しんでください．

なおこの本では，基本的で重要な命令やその使い方はすべて取り上げていますが，あらゆる場合を網羅した命令の扱いや，より高度な使い方については，別のマニュアルを参照してください．

また，FM－7を中心に書かれていますが，FMシリーズは兄弟のような関係にあるため，FM－8やFM－11についても，そのほとんどの部分は共通に利用できますし，コンピュータに対する命令の集まりであるプログラムは，別の機種でも共通に利用できます．しかし，この中の一部分には，FM－7または8あるいは11でしか利用できない使い方もあるため，このような場合にはそのつどことわり書きがしてあります．

この本は，パソコンを自分の意のままに働かせるための，もっとも基本になるBASICという，コンピュータに対する命令の与え方の方法（プログラム言語といい，FMシリーズではF－BASICという）を中心に解説してあります．

FMシリーズを，F－BASIC以外の方法で働かせることはもちろんできますが，その詳細についてはこの本では省略します．ただ，F－BASICにこれらの方法を協力させて働かせる場合については，簡単に触れることにします．

ではこれから，FMシリーズパソコンの世界に入っていくことにします．

1.1 FMシリーズの構造

F－BASICとはどんなものかを調べる前に，まずFMシリーズの内部構造は，いったいどのようになっているのかを，図解によって示します．もちろん，これらの図は理解を助けるためのイラストなので，正確な構成図については，別の資料によってください．

これらのFMシリーズパソコンは，いずれも基本的には同じ構成になっています．これらの構成図を見てもわかるように，いろいろと便利な機能が備わっており，多くの特長を持ったパソコンです。そして，そのことは，実際にFMシリーズに接し，利用していくことにより，さらに実感として体験することができるでしょう．

FM－7の構造

CPU
MBL68B09
コンピュータの中心になる情報加工工場だ
情報
情報
情報
情報

出し入れ自由なの
RAM
情報をしまっておくところよ。

ROM
コンピュータを動かすのに必要な読み出し専用の情報倉庫です。

PSG
音楽演奏の係よ。

Z80カード
Z80というCPUも使えるよ。

いろいろなカードが使えるよ。

カセットテープの係だよ。
カセットテープ

情報を印刷し出す係なのだ。
プリンタ

ミニフロッピィディスクとの情報のやりとり係だ。
ミニフロッピィ

通信回線とつなぐ係なのでーす。
音響カプラ

似 以
保 呂
辺 波
ROM
漢字をおぼえておく所だよ。

もっと大きなシステムにすることもできるぞ
拡張ユニット

関
所

CPU
CRTやキーボードの情報を加工する工場だ。
MBL68B09

RAM
情報出し入れ自由の倉庫さ。

ROM
固定情報がしまってある所だ。

RAM
CRTにうつし出す情報をしまっておく倉庫だ。

ぼくはCRTに絵を描く係だ
CRT

受付
キーボードからの情報受け付け係ですのよ
キーボード

時計係だよ。

FM-8の構造

FM-8の構造のあらましだよ。正確には基本システム図を見るんだね。
CPU
MBL6809
ここが情報を加工する工場だ。
情報
情報
情報
Z80カード
ミニフロッピィディスクの係だよ。
ミニフロッピィ
RAM
情報をしまっておくところよ。
出し入れ自由なの。
バブルカセットも便利だよ。
バブルカセット
カセットテープで情報の出し入れができるぞ。
カセットテープ
ROM
コンピュータを動かすのに必要な情報倉庫です。
読み出し専用よ。
情報を印刷し出す係なのだ。
プリンタ
通信回線とつなぐ係なのでーす。
音響カプラ
ROM
恐欲誘我
身食昼飽
肥腹睡仕
満空眠事
漢字の館だ。ひらがなやローマ字などもあるぞ
ブザー
情報
どんな情報でも持ってこーい。
ジョイスティック
各種センサ
もっと大きなシステムにすることもできるよ。
拡張ユニット
CPU
CRTやキーボードの情報を加工する工場。
MBL6809
キーボードからの情報受け付け係ですのよ。
キーボード
RAM
情報出し入れ自由の倉庫さ。
ぼくCRT係だ。
CRT
ROM
固定情報がしまってある所。
RAM
CRTにうつし出す情報をしまっておく倉庫だい。
タイム
時計係だよ。

FM-11の構造
CPU
8088
68B09E
高性能の情報加工工場だよ。
情報
情報
情報
情報
RAM
情報をしまっておくところよ。出し入れ自由なの。
ROM
漢字の倉庫だ。ひらかなやローマ字などもあるぞ。
ブザーコントロール
音を作り出す係だよ。
タイマー
時計の係でーす。
ミニフロッピィと情報のやりとりをする係だ。
ミニフロッピィ
情報を印刷しだす係だよ。
プリンタ
通信回線とつなぐ係なのです。
音響カプラ
カセットテープでも情報の出し入れができるよ。
カセットテープ
1
2
3
4
5
Z80カード
拡張メモリカード
RS-232Cカード
8"FDCカード
ハードディスクIFカード
バブルメモリIFカード
CPU
68B09E
CRTやキーボードの制御をするよ。
RAM
出し入れ自由の情報倉庫だ。
ROM
固定情報がしまってある所だ
RAM
CRTにうつし出す情報をしまっておく倉庫だよ。
ぼくはCRTの係だ。
CRT
受付
私、キーボードの受け付け係よ。
キーボード
タイム
時計係だ。

1.2 コンピュータは2進数で動く

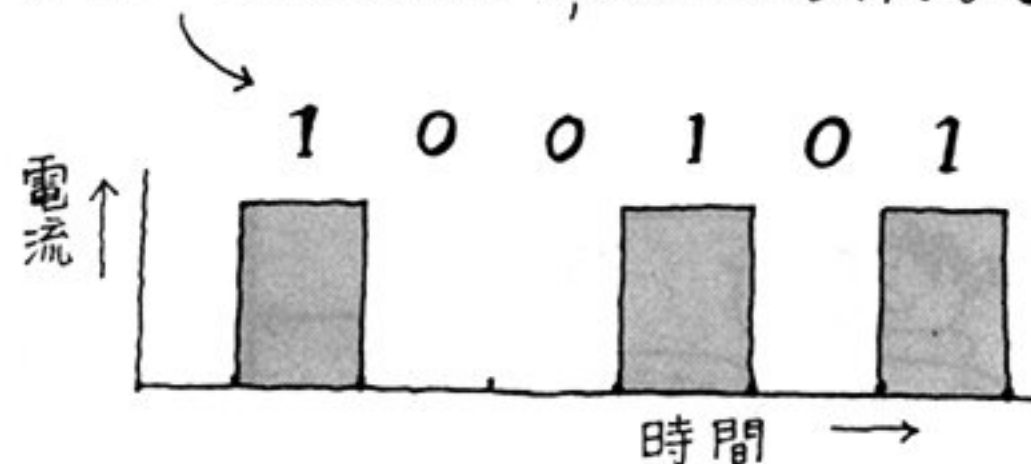
パソコンは，あの小さな箱の中に，全部集計すると，数百万個にものぼる電子回路素子が詰まっていて，その1/2～1/3はトランジスタです．

まだ半導体技術が確立されていなかった以前は，現在のトランジスタに相当する真空管が，数十本も使われていた装置というと，高度な専門技術を持った人でなくては扱えない機械であり，数百本，数千本の真空管を使った実用的な装置は，この世に存在しなかったといえます．扱いが困難なだけでなく，ぼう大な電力を必要とし，何よりも真空管の寿命が短いため，使っているより修理している時間の方が長い，という結果になってしまうからです．

しかし，今は違います．ICやLSIという小さな電子部品の中に，上述のようなばく大な量の電子部品が詰め込まれ，ごくわずかな電力で動作する，半永久的寿命の装置が低価格で作れるようになりました．この驚くべき発達は，基本的な能力の向上だけでなく，使いやすさという機能までも，たくさん取り入れることができるようになりました．

今まで人間のやる仕事と思われていた部分まで，機械が自動的にやってくれる機能を持ったのです．専門家でもない人が，この複雑な機械を自由に扱えるようになったのは，その成果です．FMシリーズは，そのような配慮が至るところになされ，驚くほど使いやすい機械になっていることは，他と比べてみればただちにおわかりいただけるでしょう．

なお，ここでいうIC（Integrated Circuit 集積回路）とは，小さな半導体（シリコン板）の上に，たくさんの回路素子をぎっしりと寄せ集めて一体化した電子部品のことで，LSI（Large Scale Integrated Circuit 大規模集積回路）とは，ICの中でも，数千～数十万個と，よりたくさんの回路素子を詰め込んだものをいいます．

◆ ところで，こんなにたくさんのトランジスタは，いったいどんな動き方をしているのでしょうか？ 簡単にいってしまうと，その多くはスイッチのような役割を果たしています．つまり電流を流したり流さなかったり，電圧を伝えたり0にしたり，スイッチのON，OFFと同じ仕事です．

コンピュータの中には，このような特殊なスイッチが無数に並んでいて，人間が指示した命令にしたがって，自動的に順序正しくON，OFFの動作を繰り返しているのです．

したがって，そのような回路を流れる電気信号も，全て流れているかいないか，つまり1か0かによって動いています．そこで，コンピュータはこのような動きに対応する2進数で動いている，といいかえることができるのです．

たとえば，キーボードのキーを押すと，そのキーのスイッチがONになり，その文字に対応する何桁かの，2進数の信号を発生する回路に伝えられるしくみになっています．

超LSIは?

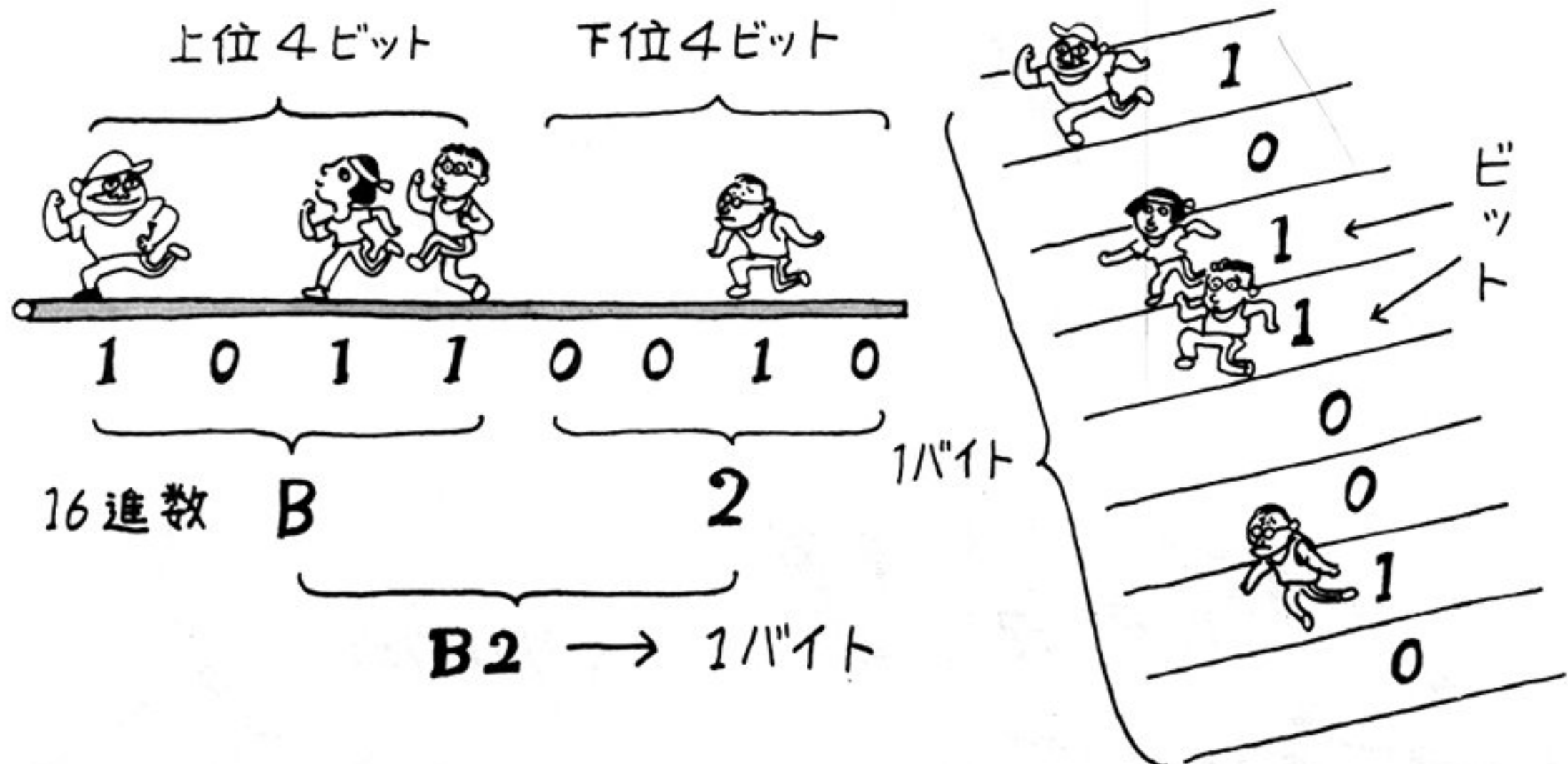
◆　コンピュータの内部で動き回る２進数は，１桁ずつ扱っていたのでは，時間がかかりすぎるため，普通は何桁かをまとめて，同時に扱う場合が多いのです．２進数の信号１桁を，コンピュータでは１ビット（１bit）といいます．たとえば，４桁の２進数なら４ビットの信号といった具合です．

そして８ビットのことを，特に１バイト（１Byte）といいます．１バイトは，コンピュータにとって，もっとも基本になる数の集まりだからです．たとえば，ＦＭシリーズでは，データや命令を，基本的には１バイト単位でまとめて扱いますし，情報がどこにあるかを示すアドレス（番地）の情報には，２バイトが利用されます．

それにしても，２進数は１と０との集まりであるため，10進数になじんでいる私達にとっては，扱いにくい数値ですね．そこで，コンピュータの分野では，10進数ほどではないけれど，２進数より扱いやすいし，しかも２進数に直すのが簡単な，８進数や16進数がよく利用されています．特に16進数は，１バイトの２進数をちょうど２桁の16進数で，うまく表わすことができるので，広く利用されます．

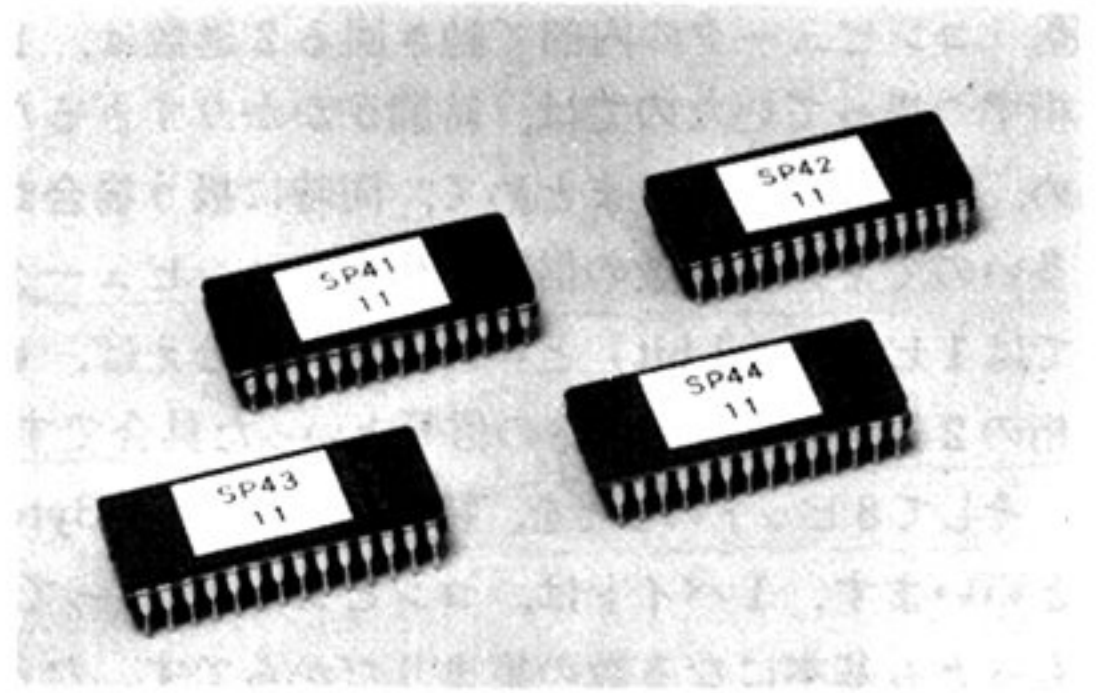
８進数は１，２，３，……７，10，11……のように，10進数の８になったら２桁の数字10になります．つまり８，９という文字を使ってはいけないわけです．また，16進数は，10進数の15までは２文字になっては困るので，表のように10進数の10に当たる数字の文字をＡとし，11以下を順にＢ，Ｃ……Ｆという文字で代用することになっています．

10進数	２進数	８進数	16進数
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16

８進数は，２進数３桁で１桁分，16進数は２進数４桁で１桁分に当たることに注意してください．

1.3 ファームウェア

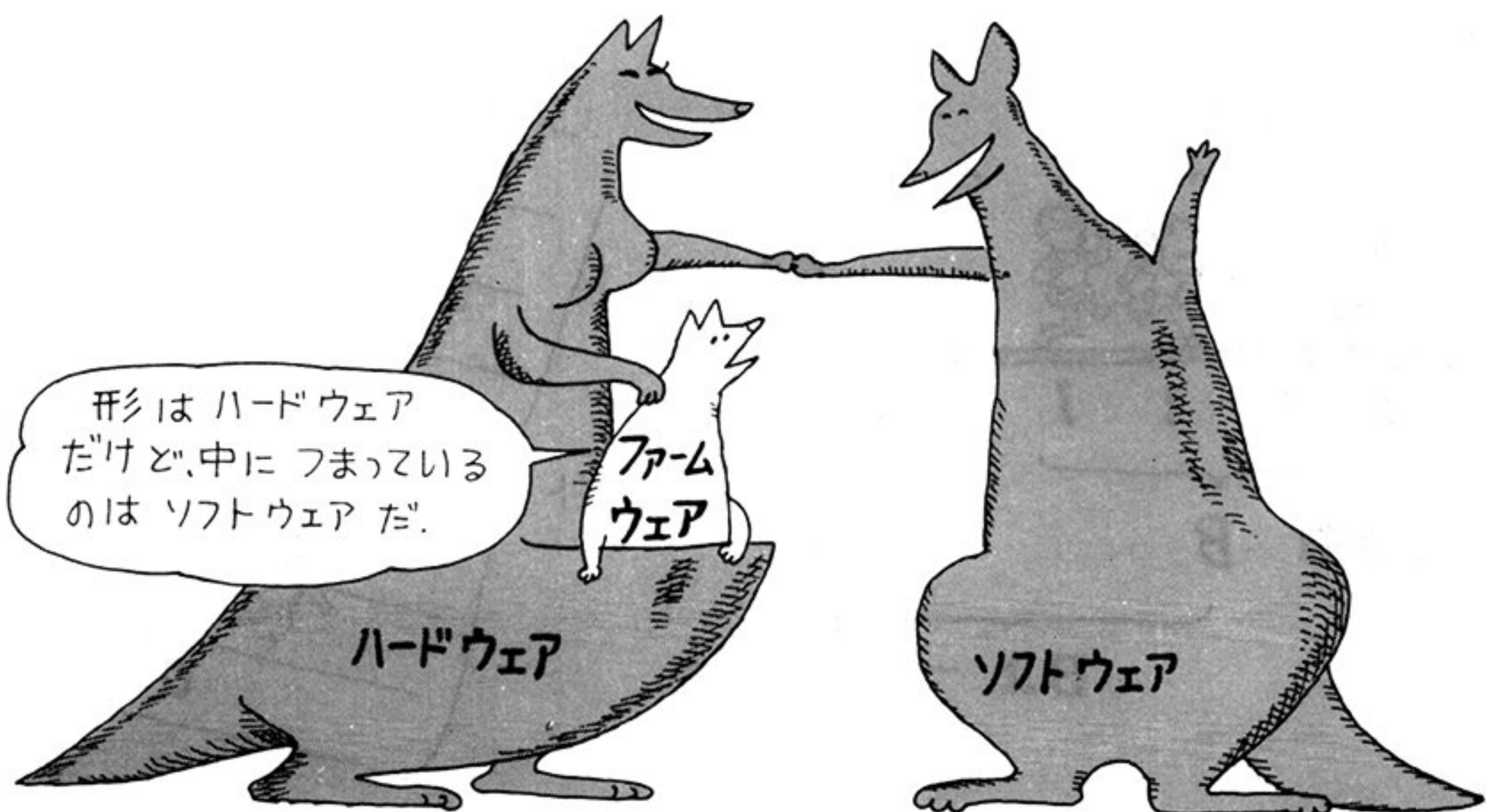
コンピュータは，機械だけあっても，ただそれだけでは何もしてくれません．機械に与える人間の命令があって初めて，その能力を発揮できるのです．一般に，この機械類のことをハードウェア，そして機械に与えるいろいろな命令のことをソフトウェアと呼ぶことは，もうご存知ですね．つまりコンピュータは，人間が与える命令どおりの仕事をする機械であって，命令にちょっとでもルール違反があると，絶対に思ったとおりの仕事はしてくれません．

よくSF小説などに出てくるように，コンピュータが人間のいうことを聞かなくなって，人間を征服してしまうなどといったことは，少なくとも現在のコンピュータでは不可能ですし，今日のコンピュータの延長上にある限り，未来のコンピュータでも考えられないことです．

コンピュータは，文字どおり徹底した石部金吉なのです．したがって，コンピュータに接するには，あとはコンピュータが適当に判断してくれるだろう，などといった甘い考え方は通用しません．与える命令は，いろいろな意味で完全無欠でなくてはならないのです．

ROM（読み出し専用の記憶素子）

2進数が16進数になったとしても，そんな面倒な数字のら列ではまっぴらだと，お考えの方もあるでしょう．しかし，その心配は無用です．コンピュータの機能の拡大は，使いやすいコンピュータへの変身を成功させました．あなたは，何も2進数や16進数で，命令を与えてやらなくてもよいのです．

この本のあとの方のページを，ちょっとめくってみてください．コンピュータに与える命令（プログラム）らしきものは，やさしい，英単語の集まりでできていますね．そこにわからない単語が書いてあっても，ちゃんと説明がありますから，今は気にする必要はありません．

コンピュータに与える命令は，人間にもわかりやすい文字の集まりでできています．もちろん，これには一定のルールがあることは，いうまでも

ありません．

しかし，それは意外に簡単です．そして，このわかりやすい言葉は，そのままではコンピュータには理解不能なため，これをコンピュータにわかる電気信号に自動的に直してしまう，いわば通訳に当たる仕事をしてくれるソフトウェア（プログラム）が用意されています．これが，これからお話を進めるF－BASIC（以下，単にBASICという）なのです．

FM－7とFM－8の場合には，コンピュータを使いやすいものにするため，このソフトウェアを数個の記憶用（ROM）LSIの中に記憶させて，取り付けてあります．このように，ソフトウェアをハードウェアの中に詰め込んだものを，ファームウェアといいます．このファームウェアのおかげで，FM－8とFM－7は電源を入れるとただちに，BASICのルールにしたがった命令（コマンド）待ちの状態になるわけです．そして，このような使いよさのためのファームウェアは，随所で利用されています．（FM－11はフロッピィディスクにBASICが入っています）

もともと，コンピュータは初めの状態では，コンピュータを働かせるのに必要な，最小限度のファームウェア（モニタ）以外，何も入っていないのが普通です．これにBASICという通訳を入れようが，それ以外の違ったプログラムを入れようが，それは使う人の自由勝手というわけです．

普通，コンピュータに記憶できるメインメモリには，一定の限界があります．たとえば，FM－8は64kバイト以内です．そこで，常時BASICにメインメモリの多くの部分を占領されていると，BASIC以外のプログラムで仕事をさせようとするとき，BASICの分だけ損をしてしまいます．

FM－7およびFM－8では，そのような場合には，スイッチの切りかえによって，BASICをメインメモリの領域から追い出し，64kバイトのほとんどを，自由に使うこともできるようになっています．（FM－11については別のマニュアルを参照してください．）

◆ コンピュータに何かの仕事をさせるときは，そのために必要なプログラムを，コンピュータの内部にある記憶装置に，おぼえ込ませておかなくてはなりません．このための記憶装置を，メインメモリ（主記憶装置，または内部記憶装置）といいます．これに対して，たくさんの情報を記録保存しておく外部にある記憶装置を，補助記憶装置または外部記憶装置といい，プログラムの進行に応じて，補助記憶装置と自動的に情報のやり取りをすることも可能です．

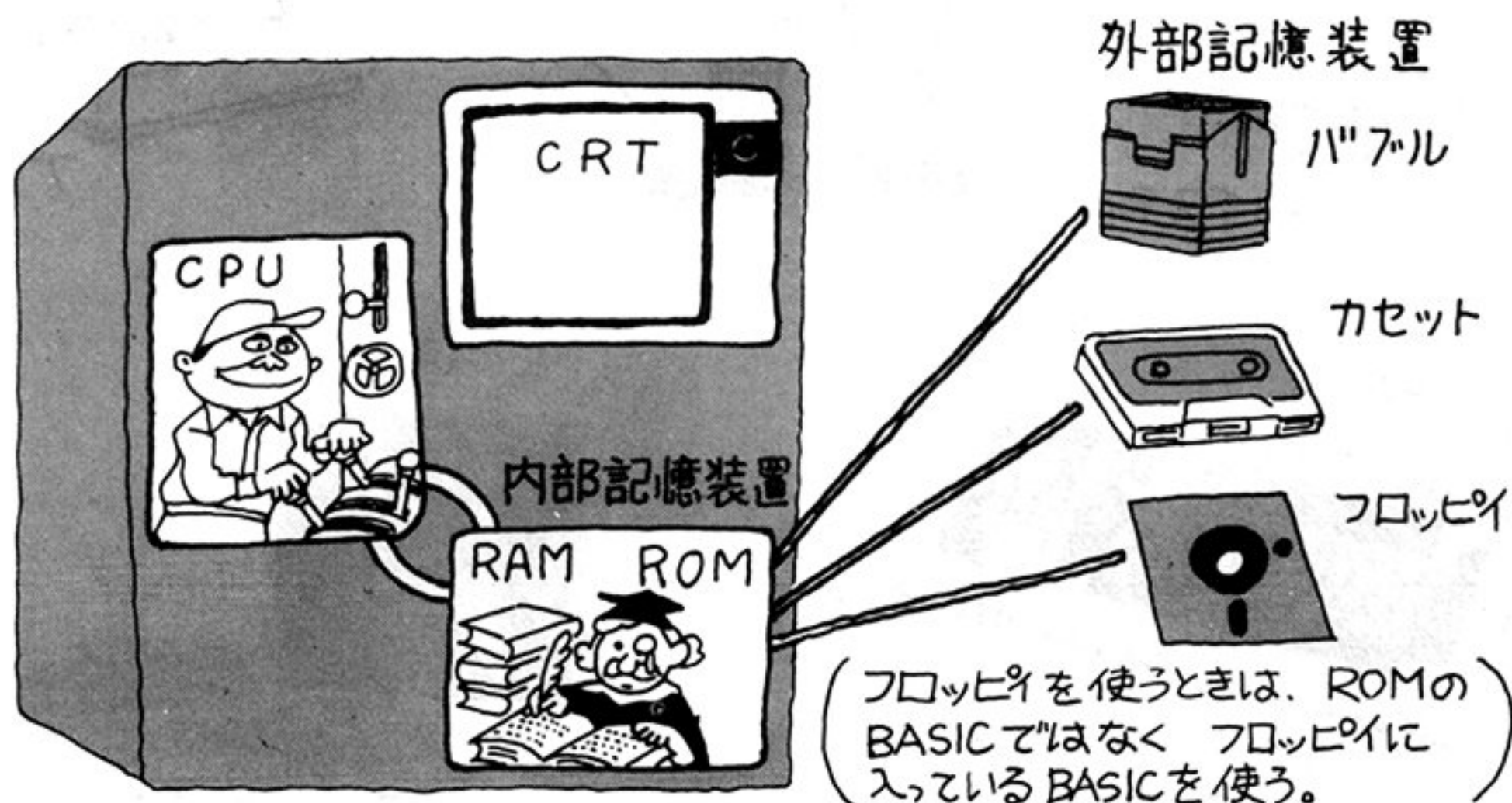
ここで，64kバイトという言葉が出てきましたが，コンピュータの分野では，2進数が基本になっているため，1k（キロ）を10桁の2進数の値，つまり$2^{10}=1024$として考えます．

1kバイト＝1024バイト

64kバイト＝65536バイト

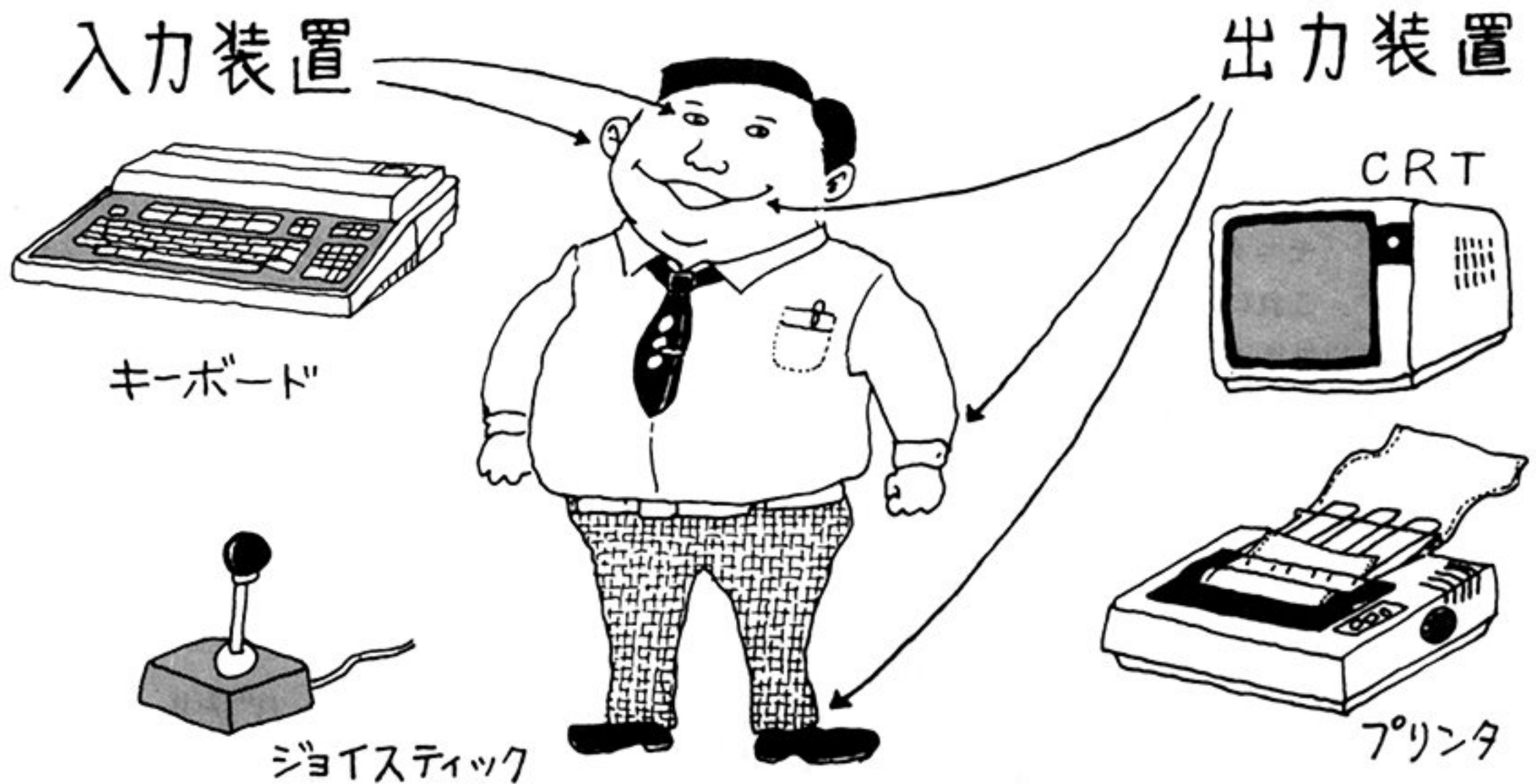
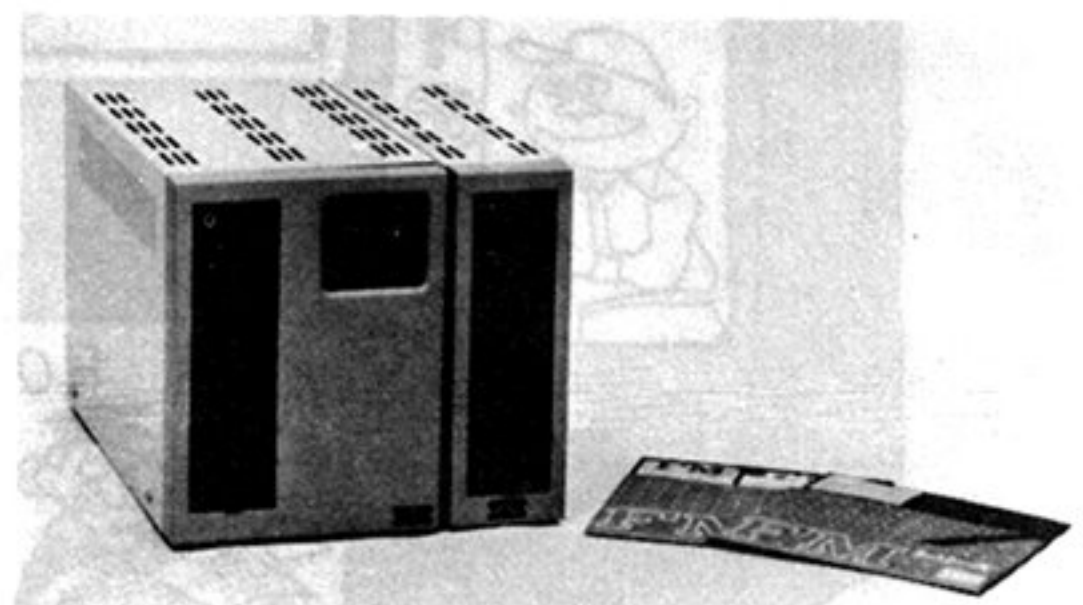
メモリは慣例にしたがって，このように書きますが，読むときはメモリーと読んでください．

1.4 入出力装置，外部記憶装置

FM－8の場合のバブルメモリホルダ部分

人間は，目で見たり耳で音を聞いたりして情報を取り入れ，神経を使って脳に伝えます．脳はその情報や，もともと持っていた情報とを合わせて，いろいろなことを感じたり考えたりします．そして，必要があれば，やはり神経を使って口を動かしてしゃべったり，手にペンを持って文字や絵を書いたり，場合によっては手足を制御し，動き回ったりします．

つまり，人間は頭脳だけあってもその働きを助ける目，耳，口，手，足などがなくては何もできないわけです．

人間のしくみを手本にしているコンピュータにも目，耳，口，手，足に相当するものが備わっています．入力装置，出力装置といわれているものがそれです．これらをまとめて，入出力装置といい，またコンピュータの周辺に置かれているので，周辺装置ともいいます．

コンピュータは，これらのハードウェアと，これをどのように働かせるかの指示を与えるソフトウェアとが，協力し合って，初めて一人前の働きをすることができるのです．そこで，これらを総称してコンピュータ・システムということもあります．

FM－7のキーボード

フロッピィディスク

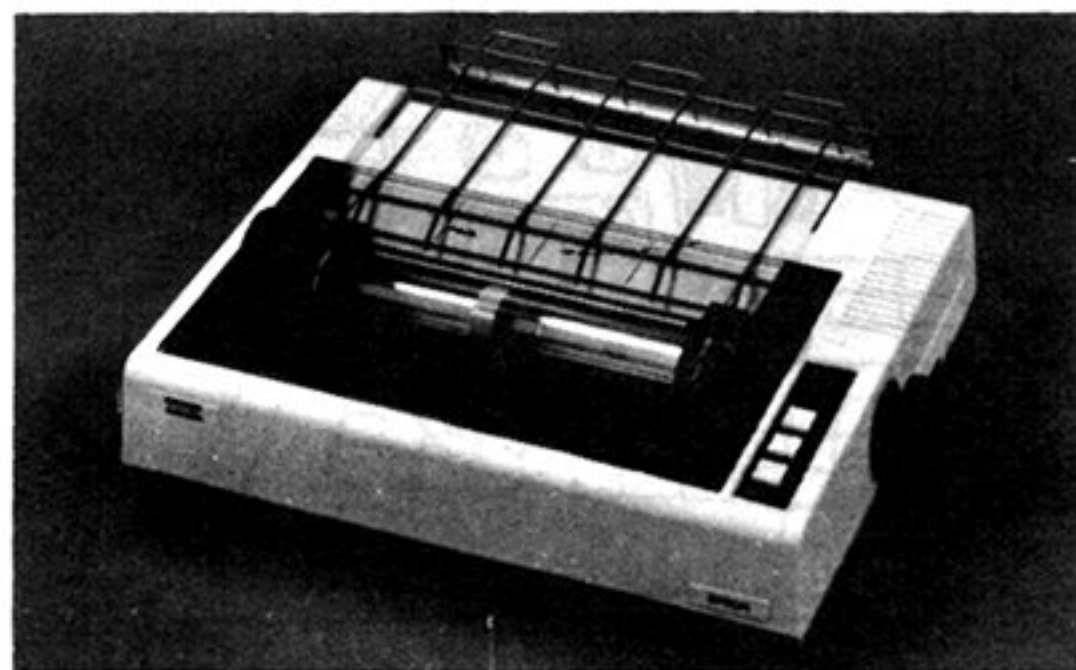
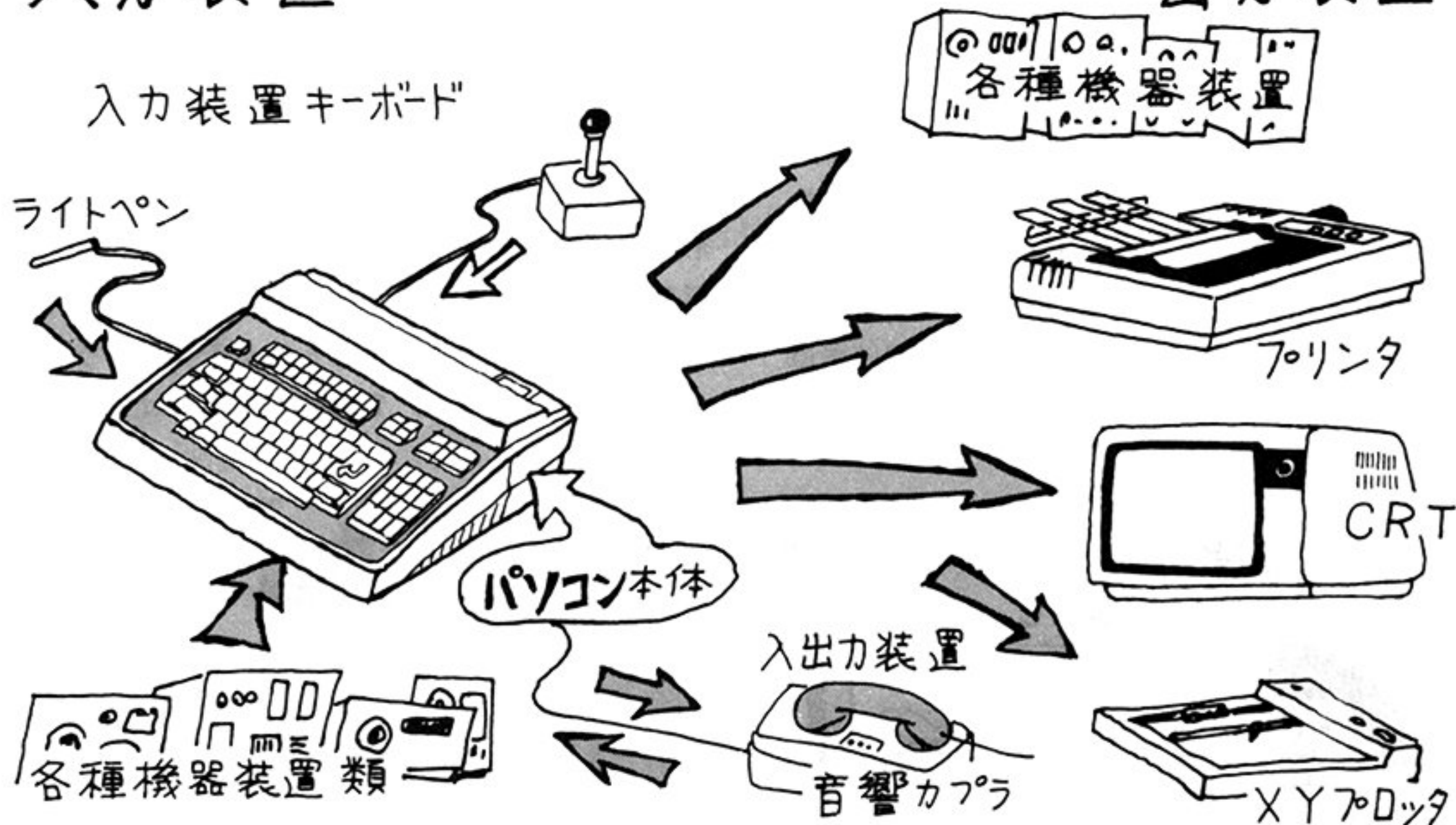
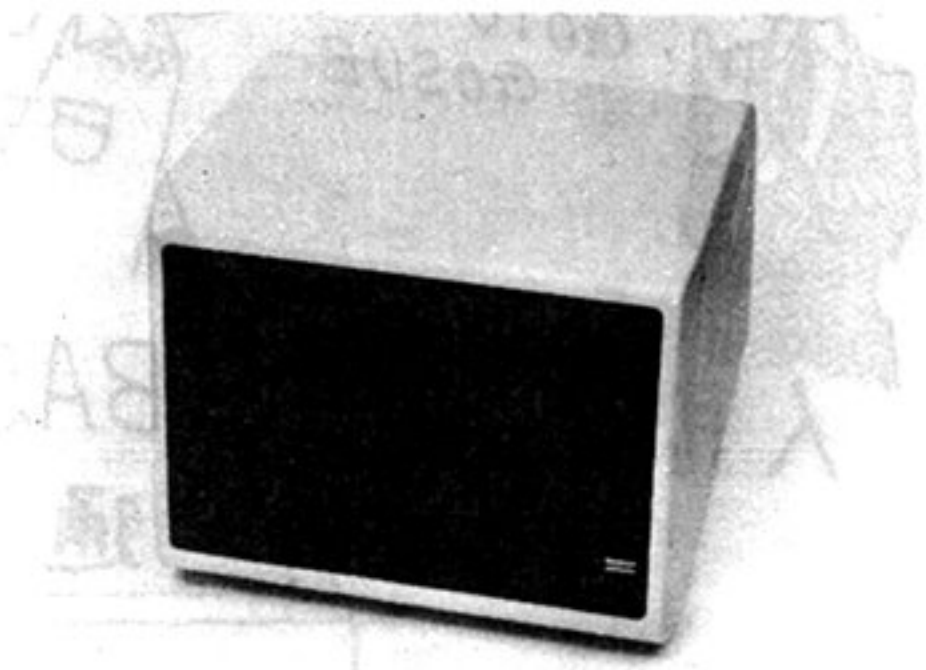
◆ 入力装置

代表的なものが，キーボードです．キーボードからコンピュータに，いろいろな情報を伝えたりします．機械の制御をしたり，コンピュータゲームをするときに便利なジョイスティック，ブラウン管に表示されている文字や記号を読み取る，ライトペンなども代表的な例でしょう．これ以外にも各種計測器やセンサなどが入力装置として利用されます．

◆ 出力装置

代表的なものがCRT（Cathode Ray Tubeの略，ブラウン管），プリンタなどです．これ以外にも入力装置同様，いろいろな種類のものが用意されていますし，また自作の出力装置を接続することも可能です．

◆ 外部記憶装置

代表的なものがバブルメモリ，カセットテープレコーダ，フロッピィディスクです．カセットテープレコーダは，一連の長い命令やデータなどを記憶しておくのに適しており（シーケンシャルアクセス），バブルメモリやフロッピィディスクは，テープレコーダと同様な働きをする以外に，記憶されている情報の中から，必要な部分だけを取り出したり，書き込んだりする，いわゆるランダムアクセスのメモリとしても利用されます．これ以外に，もっと大容量のマイクロディスクなども利用できます．

プリンタ　　CRT

1.5 BASIC

従来，コンピュータは特別のコンピュータ室に設置され，事務の大量処理，銀行のオンラインサービス，工場での機械類の制御管理，交通制御などに，社会の幅広い分野で活躍し，専門的知識を身につけた人々の手によって運営されてきました．そして今，従来程度の仕事なら軽くやってのけるコンピュータが，コンピュータ室から解放され，ごく普通の机の上にポンと置かれている時代になったのです．

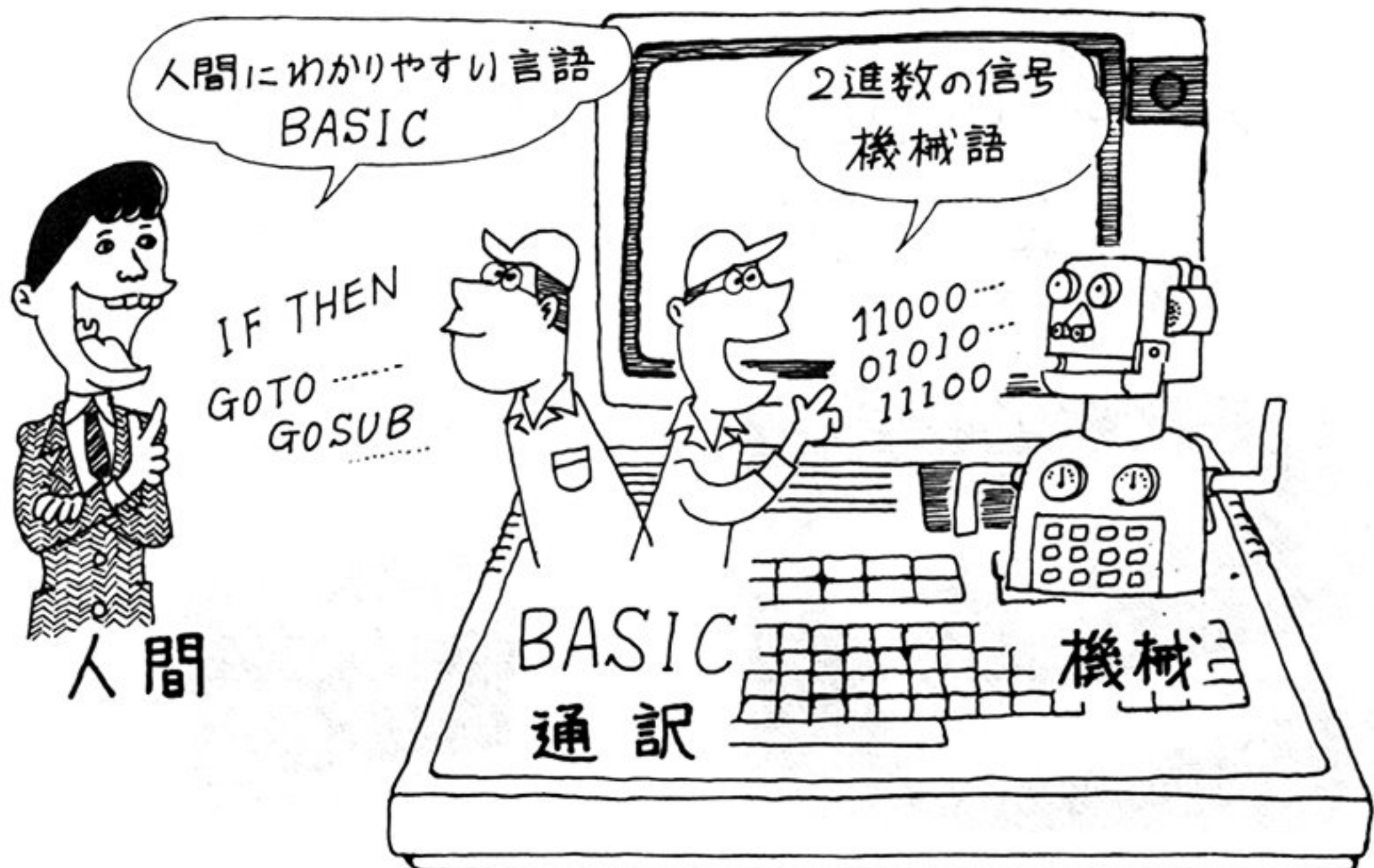
パーソナルコンピュータがそれです．そうなってくると，問題になるのがソフトウェアです．もちろん，FMシリーズに利用できるプログラムは，たくさん用意され，ちょうど音楽用のカセットテープのように，店でプログラムの入ったテープやフロッピィディスクを買ってきて，入力装置から入れると，ただちに利用できるようになっています．

しかし，パーソナルコンピュータである以上，自分専用のプログラムが必要になってくることは当然ですし，既成のプログラムを自分向けに手直ししてから使う，というような場合もありうるはずです．

```
LIST

10 REM F3.10-2
20 CLS
30 CIRCLE (300,100),50,2
40 CIRCLE (300,100),70,3
50 CIRCLE (300,100),90,4
60 GCURSOR (300,180),(X1%,Y1%),7
70 GCURSOR (300,180),(X2%,Y2%),1
80 PAINT (X1%,Y1%),6,2,3,4
90 PAINT (X2%,Y2%),5,2,3,4,6

Ready
```

BASICプログラム LIST

このような目的に合っているのがBASICです．BASICはBeginner's All purpose Symbolic Instruction Codeの頭文字を取って作られた合成語で，もともと大形のコンピュータを多くの人々がそれぞれ勝手に，まるで自分専用のコンピュータみたいに利用するTSS（タイム・シェアリング・システム）用として作られた言語です．

前述のように，ただちにコンピュータにわかる形式の言語ではなく，むしろ，ずっと人間寄りの言語です．プログラムを作り終わると，その言葉どおりメインメモリに記憶されます（もちろんそのままとはいっても，2進数の情報には変換され

ています).

そして，いよいよプログラムの実行開始命令により，プログラムの順にBASICによって機械にわかる言葉に通訳され，処理を進めていきます．逆に，機械にわかる言葉から，人間に理解できる言葉に直さなくてはならない命令に出合ったときは，もちろんそれも実行してくれます．つまり，その都度機械語と人間語との間を通訳しながら，処理を進めていくわけです．

このような処理の方法によっているので，初めから機械語で作られているソフトウェアに比べて，処理のスピードは多少遅くなります．しかし，実用上は多くの場合問題ないし，もし必要ならスピードを必要とする部分だけを機械語で作っておき，必要に応じてこの部分を呼び出して使ったり，それでもだめな場合には，別の翻訳言語（コンパイラ，アセンブラなど）を使ってBASICから離れ，処理を進めればよいのです．

BASICは，パーソナルコンピュータ用の言語として，きわめてすぐれた数々の特長があります．その第一は，何といっても命令の言葉使いや，ルール（文法）がやさしいことでしょう．また，作ったプログラムをすぐ動作させてみることができるのも特長の一つです．会話型になっているので，文法上のルール違反などがあれば，コンピュータはCRTを通じてすぐ指摘してくれます．

このようにして，作りながら途中で何回も実行してみて，気に入らないところを修正しながら，次第に完全なものに仕上げていくことができるのです．

コンピュータは，いかにすぐれたシステムであっても，あくまで人間が使用する単なる道具であり，それ以上の何ものでもありません．コンピュータはプログラムどおりの仕事しかしません．もちろん，命令以外の仕事はしないからこそ立派な道具として通用するのであって，必要以上の仕事をしたら，かえって扱いが面倒で使いものにはならないでしょう．

そのかわり，たとえコンマ一つ間違っても，「ハハーン人間様は，本当はセミコロンなのに間違えたな，直してやろう」などということは絶対にしてくれないため，一点の間違いもないプログラムを作ってやらなくてはなりません．つまり，コンピュータを生かして使うも，下手な使い方をするも，それは全くプログラムを作る人次第というわけです．

BASICは，機械に可能な範囲でプログラム作りの手伝いをしてくれます．しかし，内容をどうするかまでは，全く手伝えないのです．それは人間の仕事だからです．

1.6 コンピュータを動かす

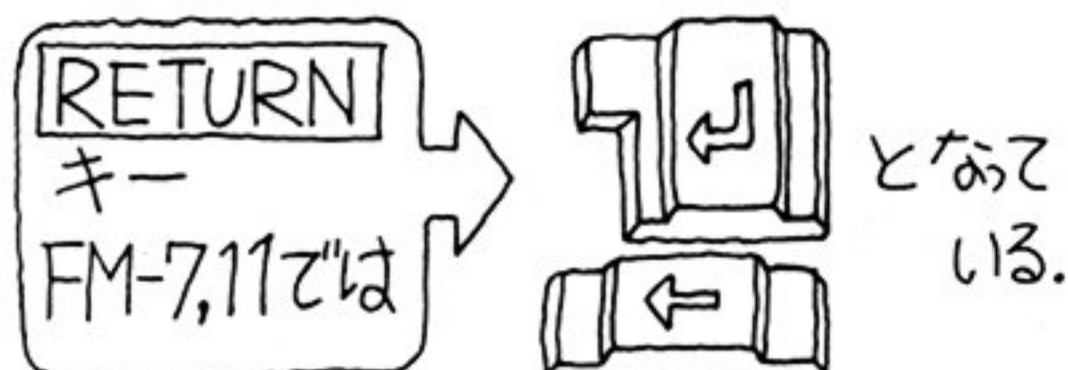
いよいよコンピュータを動かす仕事です．とにかく，電源を入れてください．周辺装置がある場合には，まず周辺装置に電源を入れ，そのあとでコンピュータ本体の電源を入れるのが正しいやり方です．切る場合もコンピュータ本体を切ってから，周辺装置を切ってください．

コンピュータの電源スイッチを入れると，赤ランプ（本当は発光ダイオードですが，この本では赤ランプとします）が点燈します．そして，CRT上に写真のような文字が出るはずです．

■の部分は点滅していますね．この■をカーソルといい，キーボードからキーインされた文字は，この位置に表示されます．そして，■は1字分右にずれます．右端一杯まで文字が詰まったら，1行下がって左端に移動するはずです．

もし，同じ文字をたくさん並べたければ，キーを押し続ければよいのです．

カセットテープレコーダに，プログラムの入っているカセットテープを入れてください．次に，カーソルを左端に置いた状態で，キーボードからLOAD "CAS0:" と入れてください．この文字の右側の方に違う文字があると，エラー（間違い）となり，正しく動いてはくれません．

エ？　大文字にならないですって？　コマンドは小文字でも正常に働きますが，大文字で表示したい場合は，左側の CAP キーを押してください．赤ランプがつき，このあとからキーインする文字は，全て大文字になります．なお，この状態で小文字を出したいときは， SHIFT キーを押した状態で，文字キーを押すと小文字になります．また，"などの記号を出すときも SHIFT キーを押した状態で，そのキーを押せばよいのです．"CAS0:" の0はO（オー）ではなく0（ゼロ）です．

電源スイッチを入れたときのCRTの表示

もしプログラム名がわかっているときは，つぎのようにする（途中に他のプログラムがあっても読み飛ばしてしまう）．

LOAD"CAS0:MANGE" RETURN

（プログラム名）

Readyが出たら
RUN RETURN
で実行を開始する．
LOADしたあと直ちに実行させるときは
RUN"CAS0:MANGE" RETURN
でもよい．これにより，LOAD完了後直ちに実行をはじめる．

■ プログラム作成時などで，用紙に鉛筆で書いていく場合文字を読み間違いなどしないために，つぎの表のように文字を区別して書く習慣になっている．

0 Ō	ゼロは 0 オーは ‾ をつけて Ō
1 i	イチは 1 アイは小文字で i
2 Z	ニは 2 似た文字のゼットは Z
5 S	ゴは 5 エスは ' をつけて S
U V	ユーは U ブイは V
T j	ティは T ジェイは小文字で j

スパイ映画では007をオーオーセブンとか，ダブルオーセブンなどと称し，ゼロとオーとを明確に区別する必要がない場合もありますが，コンピュータではそうはいきません．ゼロとオーとは明確に違うのです．そこで，コンピュータのプログラムを作る人は，このような間違いを避けるために，紙に書くとき表に示すように，文字を区別して書く習慣になっています．あなたも利用するようにしてください．

?　間違えて入れてしまいましたか？　でも，少しも驚くことはありません．たとえば BREAK キーを押すなどして，もう一度入れ直せばよいのです．あるいは， RETURN キーを押してやってもピーッという音と共に Syntax Error （文法上のあやまり）の表示が出て，あなたに再びキーインをうながします．

BREAK を使わないで，せっかく表示した文字だからこの文字を修正したい，というのでしたら，右上に並んでいる矢印のキーを押して，カーソルを移動させてください．たとえば0（ゼロ）と入れるべきところを，O（オー）と入れてしまったら，このO（オー）の文字の上にカーソルを移動させて重ねてください．そして0（ゼロ）の文字を押してください．O（オー）は0（ゼロ）に修正されます．

またLOADを， LO␣AD と入れてしまったら，この␣（スペースという）のところにカーソルを移動し，右上に並んでいる DEL キーを押してください．スペースは消えてなくなります．この操作は，スペースのかわりに違う文字，たとえばIという文字が入っていても，同様にして消してしまうことができます．LOADをLOD と入れてしまったら，やはりカーソルを移動してDと重ねてください．

次に INS キーを押します． INS キーの左側に，赤ランプが点燈したことを確認した後，Aキーを押すと，Dという文字の左側にAという文字が割り込んできます．なお， INS の赤ランプがついたままにしておくと，いつまでも割り込みたがり屋の性質を持ち続けるため，もう一度 INS キーを押して，赤ランプを消しておいてください．

さあ，CRT 上の文字は正しくなりましたね．では，カセットテープレコーダの再生キーを押してみてください．テープレコーダは動きませんね．ではLOAD"CAS0:"を入れた後， RETURN キーを押してください．レコーダは動き出します．ただし，この状態はリモコン用のコードを付けた場合の話です．

なお，リモコンがついている場合には，モータの一時停止を解除するときはMOTOR ON RETURN とすればよく，またMOTOR OFF RETURN でふたたびコントロールできるようになります．

RETURN キーを押すことは，CRT上に表示されている命令をコンピュータに伝え，その命令を実行させることを意味します．ただCRT上に表示されているだけでは，コンピュータは，まだ実行せよという意味には受け取っていないのです． RETURN キーを押して初めて実行せよ，という命令を受け取ります．

RETURN キーを押すと，コンピュータは，Searching という表示を出すはずです．これは，「探しているよ」という意味です．そしてカセットテープが回り，目的のプログラムを発見すると「FOUND:　テープの名前」が出て，テープレコーダに記憶しているプログラムをコンピュータに伝えます．コンピュータは記憶し終わるとReadyと表示し，完了したことを告げてくれます．

さあ，これでテープの中に収められていたプログラムは，ただちに実行できる状態になりました．実行の命令は，RUN RETURN です．

この状態では，テープはLOADを完了したままの状態です．もし巻き戻しておきたい場合は，コンピュータからのリモコンを解除してから巻き戻してください．また，次のプログラムを入れたい場合は，巻き戻さずに，もう一度上記の手順で実行してください．コンピュータは，前におぼえ込んだプログラムをすっかり忘れてしまい，新しいテープのプログラムをおぼえ込みます．

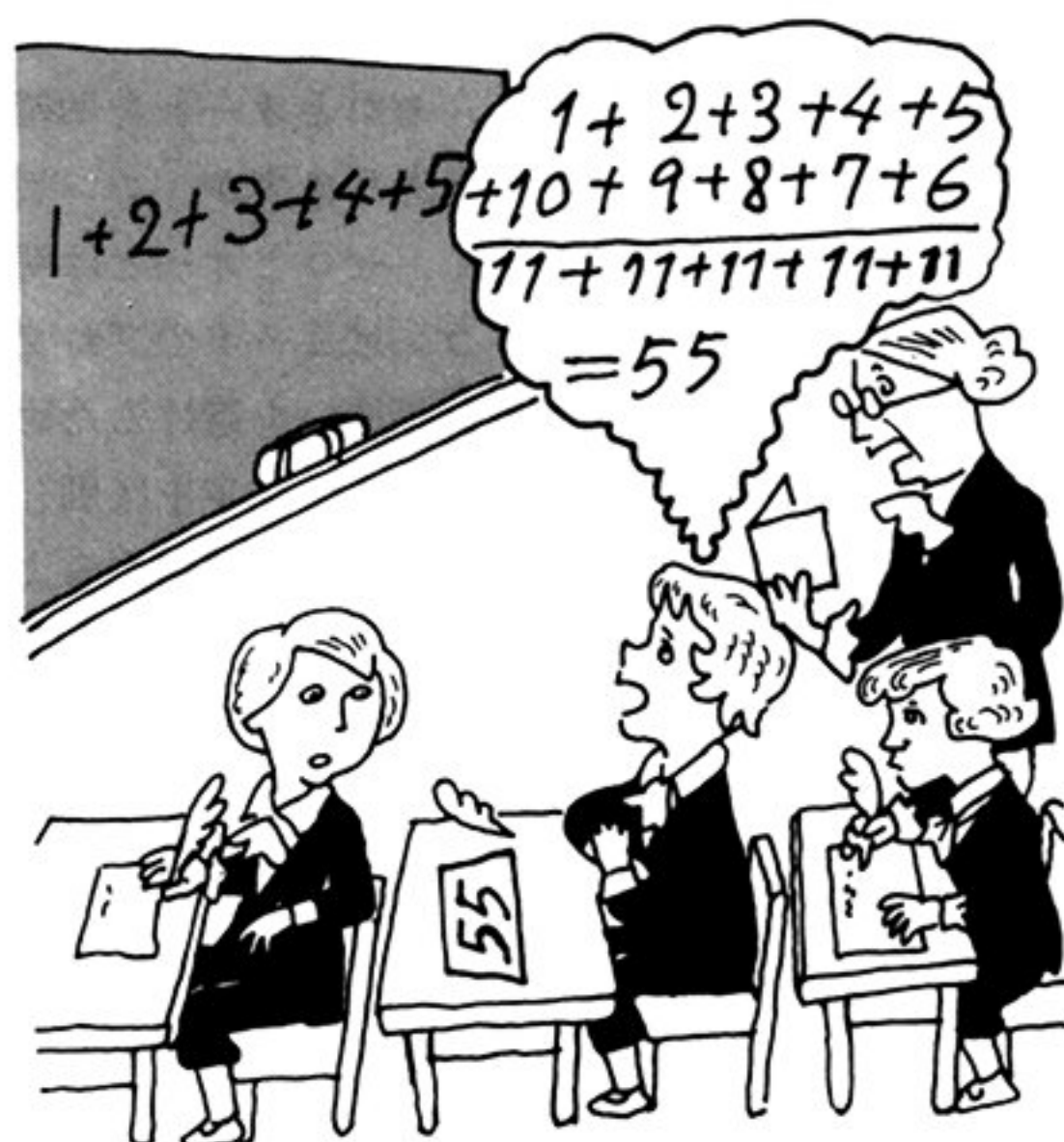
また，巻き戻さずそのままにしておき，次に使うときは，LOAD RETURN を入れた状態で，巻き戻すキーを押してから，再生キーを押してやってもよいのです．なおLOADしたあと，自動的にすぐ実行に移したいときは，LOAD "CAS0:" のかわりにRUN "CAS0:" としてください．

1.7　三つのモード

前述のように，パソコンにフロッピイディスクなどの記憶装置からLOADし，働かせてやると，ただちに実用に使えるプログラムは，これからもたくさん市販されるようになるでしょう．あなたが欲しいと思うプログラムを，全部あなた自身の手で仕上げてしまうということは，決して楽な仕事ではありません．必要に応じて，人の作ったプログラムを入手することも必要な仕事でしょう．

しかし，自分で自由に使いこなすというところに，パーソナルコンピュータの値打ちがあるわけですから，人手にたよったものだけでなく，自分自身で作ったり，あるいは既存のプログラムを，別の仕事用に改造したりすることが必要になってきます．

そのためには，BASIC言語の基本をしっかり身に付けなくてはなりません．BASICはやさしい言語です．一つ一つ順を追っていけば，必ずだれにも自由に使いこなすことができるようになるはずです．ただ，仕事をやらせる手順は，前述のように，あなた自身がおぼえていかなくてはなりません．

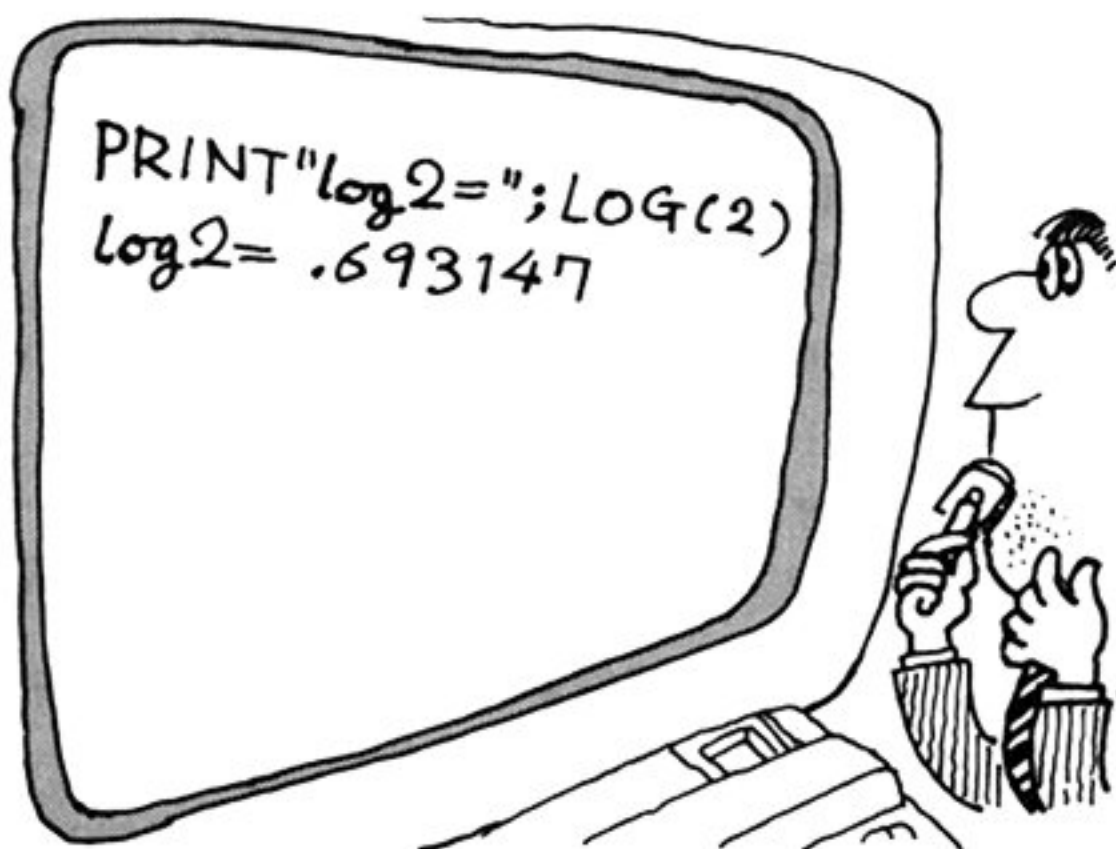
このように，どんなふうにコンピュータに仕事をやらせるかの手順のことをアルゴリズムといいます．アルゴリズムにはいろいろなテクニックがあり，いかに上手に仕事を運ばせるかは，やはりいろいろなプログラムを実際に作ってみたりして体験し，上達しなくてはなりません．

たとえば，2×3という計算を2＋2＋2としても結果は同じですね．1＋2＋3＋……＋9＋10を，人間なら11×5とすれば楽だと，考え付くことができます．しかしコンピュータには，こんなことはできません．つまり，まずいいアルゴリズムを考え出すことが大切であり，これができればあとは，BASIC言語を上手に使い分けてプログラムを作っていけばよいのであって，時間との闘いになります．

このようにして上手な作り方をすると，処理時間が短くなったり，プログラムの長さが何分の1かになったりすることもあります．前置きがすっかり長くなってしまいました．

コンピュータに与える命令のし方には三つの種類があります．直接モード，間接モード，ターミナルモードです．それぞれどのように違うのか調べてみましょう．

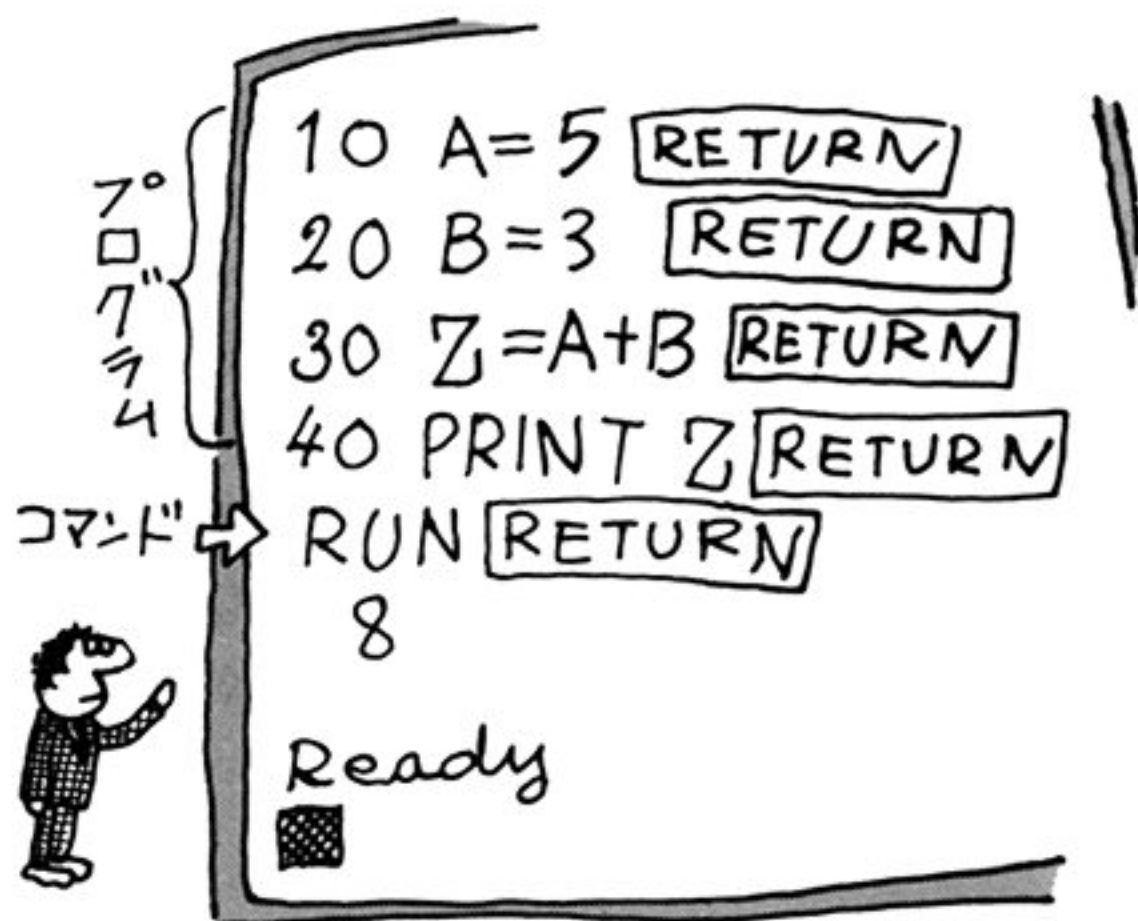
◆ 直接モード

BASICに使われる命令用の言葉で，いきなりキーボードから入れてやる方法です．命令用の言葉には，コマンドとステートメントとがあります．コマンドは，前述のようなLOADとかRUNのような，コンピュータを直接制御してやるための命令のことをいい，ステートメントとは，ある一つの仕事をさせる命令文のことです．たとえば，A＝B＋CやIF A＝0 THEN 10（もしA＝0なら10行目に飛んで行け！詳細は後述します）などもステートメントの一つです．RETURN キーを押すと，ただちに実行します．

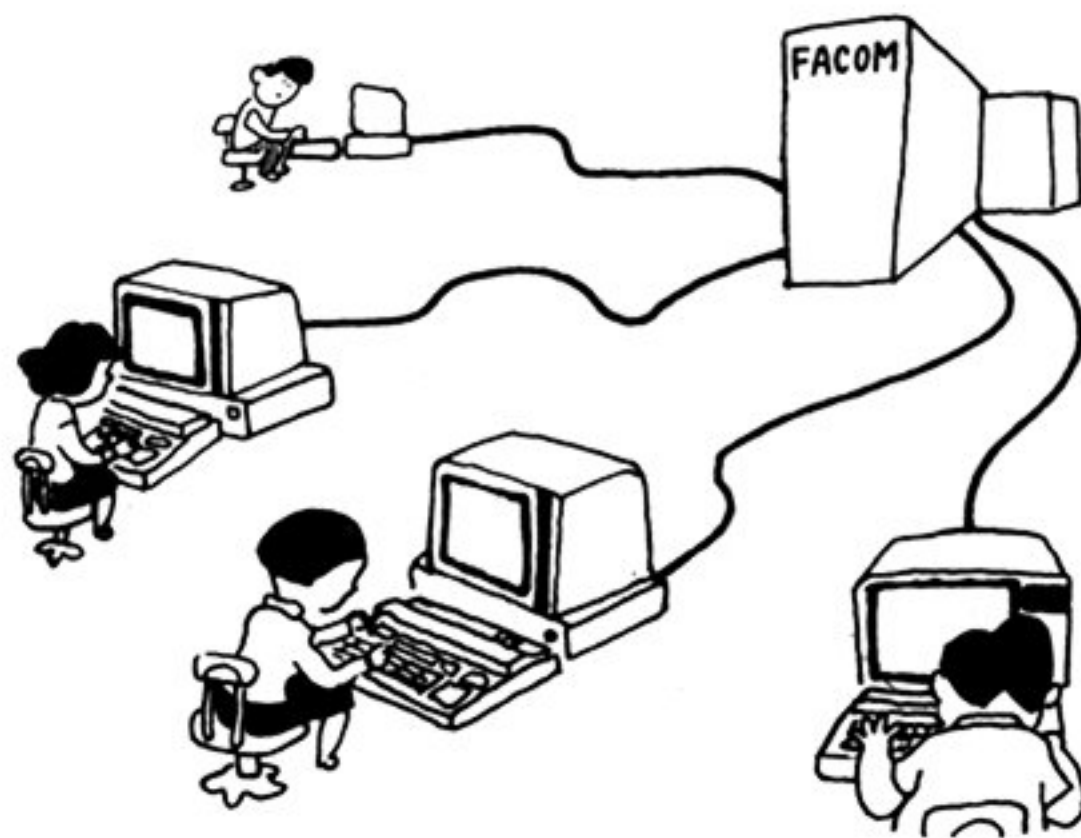
なお，＝や＋などのように，キーの表面上側に書いてある文字を出したいときは，SHIFT キーを押しながら目的のキーを押してください．直接モードによる命令文の長さは，最大255文字です．コマンドは20数種あり，ステートメントに使われる命令語もたくさんありますが，いずれも必要にしたがって，説明を進めていきたいと思います．

◆ 間接モード

何行にもわたるコマンドやステートメントでできた長い命令です．各行の先頭には，必ず行番号を付けてやらなくてはならず，行番号が付けられると，自動的に間接モードで働くようになります．この場合は，直接モードとは異なり，RETURN キーを押しても，ただちに実行に移されることはなく，各行が完了するたびに RETURN キーを押すことにより，CRT上に表示されたその行の命令文が，メインメモリに記憶されるという仕事をします．

もちろん，メインメモリに収容されても，あとで修正してやることは可能です．そして，このようにしてでき上がったプログラムを実行に移すときは，RUN RETURN によって始まります．

なおこの場合も，1行につき最大255文字まで，行番号は0から63999までの整数が使えます．

◆ ターミナルモード

パソコンを，他のコンピュータの端末装置として使う場合であり，TERMというコマンドを入れてやると，このモードになります．なお，いたずらにTERMと入れると，制御が外部に移ってしまい，動きがとれなくなるため，必要な場合以外は利用しないでください．

なお，端末装置とは，一般に離れた場所に置き通信回線で接続して利用する装置をいいます．たとえば，銀行のオンラインシステムでは，中央にある大型コンピュータと通信回線で，各支店に置いてある端末装置とを接続し，預貯金業務を全国的に一括してできるようになっています．

1.8 直接モード

さあ，それではあなたの作った命令によって，コンピュータを自由に働かす仕事に取りかかりましょう．初めは直接モードです．この命令は，記憶しておき，あとで何回も利用しようというわけにはいきません．このような場合には，たとえ1行の短い命令であっても，後述の間接モードによらなくてはなりません．

CRT に何かを表示せよ，という命令は PRINT です． PRINT のかわりに？を入れても同じ結果を得ます．では， PRINT を使ったいくつかの例を調べてみましょう．

```
PRINT 3+5        RETURN
 8

Ready
```

Ready は次をどうぞ，という意味です．

コンピュータは，3＋5の計算をし，その結果を表示してくれました．全体が255 文字以内なら，もっと複雑な計算も，もちろんやってくれます．しかし，よく注意してみてください．答えの8の前に1字分のあき（スペース）があります．なぜでしょう？ 試みに＋を－に変えて同じことをやってみましょう．答えは負数になるはずです．

```
PRINT 3-5        RETURN
-2
```

これでおわかりですね．前述の1字分のスペースは，正，負の符号が入る場所だったのです．ただ，＋符号の場合はいちいち表示するとわずらわしいので，私達の慣例にしたがって，省略されているわけです．

では，次の例はどうでしょう？

```
PRINT"3+5"       RETURN
3+5
```

今度は，計算せずに〃〃（ダブルクォーテーションマーク）の中の文字を表示しました．〃〃でくくると，コンピュータはその中を単なる文字（または記号）としてしか見ず，そのとおり表示してきます．このような文字の並びを文字列（またはストリング）といいます．ただし，〃〃でくくった中では〃は使えません．文字列の終わりを示す〃と判断してしまうからです．

```
PRINT"3*5" RETURN
3*5

A=3 RETURN
B=5 RETURN
PRINT A*B RETURN
 15
```

255 字以内なら，別の仕事を同時にやらせることも可能です．この場合には，分離記号：（コロン）を使います．

```
PRINT"3+5=":PRINT 3+5
                    RETURN
3+5=
 8
```

初めの PRINT 文は文字列の表示，分離記号のあとの PRINT 文は，実際に計算をさせるステートメントです．もし，答えの8も3＋5＝のあとに続けて書かせたければ；（セミコロン）を使います．

```
PRINT"3+5=";:PRINT 3+5
3+5= 8              RETURN
```

もちろん，8の前の1字分のスペースは本来なら＋符号が入るべき場所ですね．

コンピュータでは，変数というものをたくさん使い分けます．数学の問題を解く場合には，変数が三つも四つも出てくると四苦八苦，頭が痛くなってしまいますが，コンピュータの変数は少しも驚くことはありません．後述するように，数字や文字列を一時的にしまっておく箱であり，他の数字や文字列と区別するために，貼ってあるラベルの名称に過ぎないからです．

もし変数名をAとすると，A＝3＋4あるいはA＝A＋1のように，まず変数名であるAを先に書かなくてはならない約束になっています．エ？ A＝A＋1は，理屈に合わないではないかですって？ イイエ，これでよいのです．左側のAは変数名，右側A＋1は，それ以前のAという箱の中に入っていた数字に1を加えよ，という意味だからです．

たとえば，今まで2という数字がAの箱の中に入っていたら，この新しい命令により，2＋1つまり3という新しい数値がAの箱の中に入ります．数学でいう変数とは少し違った考え方ですね．では，実際にいくつかの例を調べてみましょう．

```
A=3    RETURN      (1)
B=5    RETURN      (2)
PRINT  A+B         (3)
```

この操作のあと RETURN キーを押すと，もうおわかりですね．8と表示してくるはずです．コンピュータは，(1)の操作でAの箱には3，(2)の操作でBの箱には，5の数字が入ったことを，ちゃんと記憶していてくれたわけです．では，これに続いて，

```
A=A+3  RETURN
```

と入れてみて，もう一度(3) RETURN を繰り返してみましょう．結果は11と出ましたね．もう一つ，現在Aの箱の中には，6が入っているはずです．では，まだ何も定義していない変数Cと加えてみたらどうなるでしょうか？

```
PRINT  A+C
 6
```

Aの値と同じですね．つまり，初めの状態では，変数の中に何も入っていない，つまり0であることが，これでおわかりいただけたはずです．

これらの話は，もちろん直接モードだけではなく，間接モードの場合にも通用します．

直接モードは，間接モードのプログラムを作っている途中，あるいはプログラムの実行中など，いろいろな場合に利用されます．

たとえば，間接モードで作っているプログラムを調べたいとき，LIST RETURN と入れてやると，プログラムが行番号の小さい方から順に，次々とCRT上に写し出されてきます．このような入出力装置とのやり取りは，もちろん間接モードでも実行できますが，直接モードで指示する場合が非常に多いのです．

試しに，次の命令文をやってみませんか？

```
FOR X=1 TO 10000:
  PRINT"A";:NEXT X
```

RETURN キーを押してやると，Aという文字を10000回CRT上に表示してくるはずです．エ？ いつまでも同じことをやって止まらないって？ では BREAK キーを押してください．次のような表示が出て止まります．なお，ここに出てくる命令の詳細は，あとでくわしく説明します．

```
AAAAAAAAAAAA-------A
AA-----A
Break
Ready
```

①FOR X=1 TO 10000
②PRINT"A";
③NEXT X

Xが10000lになるまで①にもどれという意味．

①と③との間にある仕事(②)を10000回繰り返し実行せよという意味．②が終るたびにXは1ずつ増加し10000になるまで続く．
；は隣に続けて描け，という意味．

1.9 間接モード

行番号を付けたプログラムの場合，コンピュータは，通常行番号の小さい順に実行していきます．もちろん途中に，違った行番号のところに進め，などという命令が入っていれば，それにしたがって実行します．

そこで，BASIC でプログラムを作っていくときは，一般に行番号を，たとえば10飛びに10，20，30，……のように付けてやる方法がとられます．もし，途中に別のプログラムを追加しようとするような状態が起こったとき，たとえば行番号を15に取ってやると，コンピュータは，10，15，20，……の順に実行してくれるからです．

行番号をいちいち入れるのが面倒な場合，AUTO というコマンドを入れてやると，コンピュータは行番号を自動的に発生してくれるのでたいへん便利です．この機能を停止させるには，BREAK キーを押してやるか，次の行番号が表示されてから，RETURN キーを押してやればよいのです．

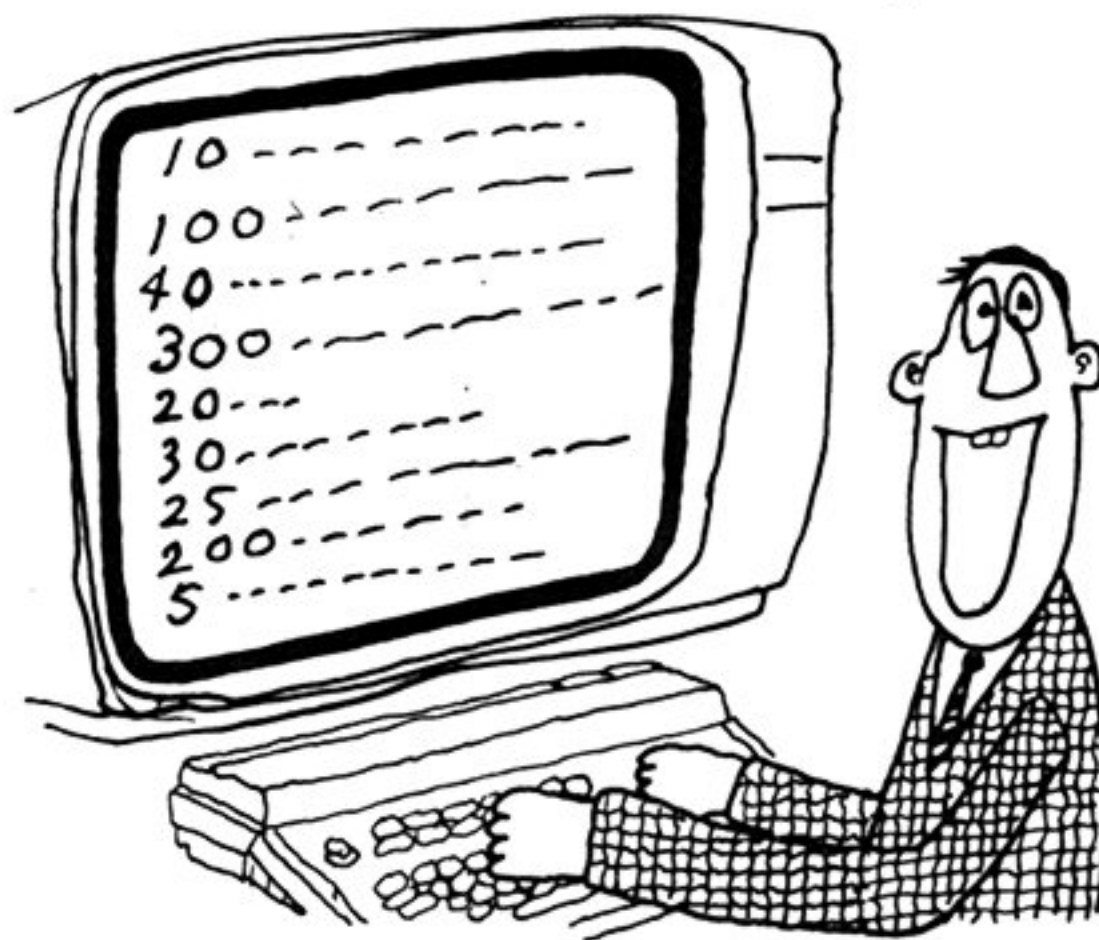
AUTO [[行番号] [，増分値]]

AUTO だけの場合は，行番号は10から，増分値が10で，つまり10，20，30，……のように行番号が表示されますし，AUTO 100，5 と入れてやると100，105，110，115……のように行番号が発生してきます．

なお，すでにプログラムが入っている場合には，AUTO だけではなく，新しく始める行番号を入れてやらないと実行してくれません．

では，一度 CLS キーを押して画面をきれいにし，次のように入れてみてください．

```
10 LET A=10      RETURN
20 LET B=15      RETURN
30 LET C=A+B     RETURN
40 PRINT C       RETURN
```

もう意味はおわかりですね．LET は，右辺の式の結果を左辺の変数に代入せよという意味ですが，省略してもよいことになっているため，以後

この本では省略することにします．なお，コンピュータでいう式とは，実際に計算式になっている場合や，上述のように単に数値だけの場合もあります．そのほかにもいろいろありますが，ここではふれません．

間接モードでは，RETURN キーを押しただけでは，実行してくれません．各行のあとに RETURN キーを押したのは，その行をメインメモリの中に記録させるための操作でした．

間接モードで実行に移すにはRUN RETURN の操作が必要です．よくコンピュータを走らせるといいますが，RUN からきた言葉です．本当に走り出したら困ってしまいます．

さて，このプログラムを走らせてやると，結果は25と出るはずです．エラーの表示があったり，正しい答えを出してくれないときは，きっとどこかに間違いがあるからです．よく調べてみてください．そして，間違っていたら，新たに間違っていた行番号を入れ，正しいステートメントによる行をキーインし，RETURN キーを押してください．今度は正しく修正されたプログラムに置きかえられているはずです．

行番号

```
1 CLS:PRINT"** ウリアゲ ヒョウ **":FORI=1TO40:PRINT"—";
:NEXT:LOCATE0,10:FORI=1TO40:PRINT"—";:NEXT:LOCATE
10,1:PRINT"┬":FORI=1TO8:J=J+1:LOCATE10,J+1:PRINT"
|":NEXT:LOCATE10,10:PRINT"┴"
```

```
10CLS
20 PRINT"** ウリアゲ ヒョウ **"
30 FOR I=1 TO40
40 PRINT"—";
50 NEXT I
60 LOCATE 0,10
70 FOR I=1 TO40
80 PRINT"—";
90 NEXT I
100 LOCATE 10,1
110 PRINT"┬"
120 FOR I=1 TO8
-------------------
```

これでも立派なプログラムだよ．行番号1の1行だけのプログラム．これでもまったく問題ないが，メモリに余裕があれば何行にもわたるわかりやすいプログラムづくりをするべきだよ．上例のようなプログラムは，わかりにくくて仕方がないね．

この方がわかりやすいし，修正も簡単ね．

念のために，プログラムをもう一度 CRT 上に表示させてみてください．LIST RETURN でプログラムを表示します．間違いのあった行は，たった今入れた正しい行に直っているはずです．一番最後に入れたプログラムが，常に新しいプログラムとして保存されるのです．

このことを逆に利用し，たとえば10 RETURN と押してやると，行番号10には何も入っていないことになるため，消えてしまいます．行番号だけ取ってあとに何も入れないでおくと，行番号も消えてしまうように作られているわけです．

NEW RETURN と入れてみてください．これはおぼえ込んでいる BASIC によるプログラムを，全部忘れてしまいなさい，という意味のコマンドです．LIST RETURN を入れても，プログラムは表示されないはずです．NEW のあと，次のプログラムを入れてみてください．

```
10 A=10:B=15:C=A+B:PRINT C
                                RETURN
```

RUN により，前と同じ結果を得ることができるはずです．

このように1行であっても，間接モードであることには違いありません．1行に並べてしまうことのできる長さは，最大255文字です．ただし，マルチステートメントをいたずらに利用すると，修正が面倒であり，またあとでのチェックも困難になるため，余裕があればあまり使わない方が無難でしょう．なお，マルチステートメントにすると，処理スピードは少し速くなります．

どんなプログラムがメインメモリに入っているかを知るには，LIST のコマンドが使われます．そして長いプログラムの場合，次々に表示が続くため，途中で止めて調べたいときは，BREAK キーを押せば停止します．また，部分的に表示したいときは，次のようにします．

LIST	全部のプログラムが表示される．
LIST 70	行番号70のプログラムが表示される．
LIST−80	行番号80までのプログラムが表示される．
LIST 90−	行番号90以後のプログラムが表示される．
LIST 100−200	行番号100から200までが表示される．

なお，プログラムは，小文字でキーインしてやっても問題ありません．このあと，LIST コマンドでもう一度表示させてやると，大文字に自動的に直されています．

1.10 スクリーン・エディタ（その1）

キーボード上のキーの使い方については，必要に応じて説明しますが，ここでは，プログラムを作る上で知っておくと便利なスクリーン・エディタ，つまりCRT上での文字の追加，削除，修正などをするときのキー操作について述べます．なお，ここで示した以外のキーの操作については，2.17で説明します．

高速でタイプするときは両手をフルに利用する方がいいよ．そのときは各指で下図のキーを押すようにするとよい．キーを押さないとき、人さし指がFとJの上にくるようにするのがいいのだ。

下図は，FM-7，FM-11のキーボードの例だよ．

キーボードの例（FM－7のキーボード）

◆ どの文字でもよいから，長くキーを押し続けてみてください．その文字が1字表示され，ちょっと間をおいてから今度は，次々に表示され続けます．特に，グラフィック図形など作るときに便利な使い方です．

◆ カーソル移動キー（矢印のキー）を押し続けてください．今度はカーソルが矢印の方向に，画面の端から端まで何回も移動します．

◆ 文字が1列に並ぶ中にカーソル移動キーを停止させ，［DEL］キーを押してください．カーソル移動キーと重なった文字が消えてしまい，その文字の右側に並んでいる文字が，一文字ずつ左側に引き寄せられます．さらに押し続けていると，まるでカーソルに吸い込まれるように，右側の文字が引き寄せられ，1字ずつ消えていきます．なお，DELは英語のdelete（削除）の略です．

◆ ［INS］キーを押すと，キーの左側に赤ランプがつき，このInsert（そう入）の機能が働いていることを示しています．この機能が働いているとき，文字のキーを押すと，カーソルと重なっている文字から右側が，全部1字分右側に移動し，その間にキーインした文字がそう入されます．プログラムの途中に別の命令を追加するときなど，たいへん便利です．この機能を中止したいときは，もう一度［INS］キーを押してください．赤ランプが消えて正常に戻ります．

◆ 今度は，［SHIFT］キーを押しながら矢印のキーを押してみてください．やはり画面上で，カーソルが特別の動き方をします．

［SHIFT］+［↑］，［SHIFT］+［↓］

［↑］の場合は下から上に向けて，［→］の場合は上から下に向けて順に，並んでいる文字の先頭にカーソルを移動させます．途中にスペースがあると，その右側の文字の先頭にも移動してくれます．

［SHIFT］+［←］，［SHIFT］+［→］

カーソルがどこにあっても，カーソルは左端に移動し，矢印の方向に動きます．

10　X=A*B*C*D
20　Y=■←カーソル

この状態で［DUP］キーを押すと

10　X=A*B*C*D
20　Y=A*B■ ←カーソル
　　（8文字）

つまり，左から8文字ごとに，上の行と同じ文字をつぎの行に写し出す．

12345678901234567890
■←カーソル

この状態で［DUP］キーを2回押すと

12345678901234567890
1234567890123456■
（1回目）（2回目）

なお，これらの仕事は，矢印のキーによらず，次のように［CTRL］キー（Control　キー）を使っても可能です．

［←］=［CTRL］+［］］

［→］=［CTRL］+［¥］

［↑］=［CTRL］+［^］

［↓］=［CTRL］+［−］

［SHIFT］+［←］=［CTRL］+［B］

［SHIFT］+［→］=［CTRL］+［F］

［SHIFT］+［↑］=［CTRL］+［Y］

［SHIFT］+［↓］=［CTRL］+［Z］

FM-7のキーボード

（FM-11の場合、キーの配置は異なるが　機能は同じ。）

BREAK（ブレーク）キー．
コンピュータの実行を途中で中止する。FM-8は STOP になっている。

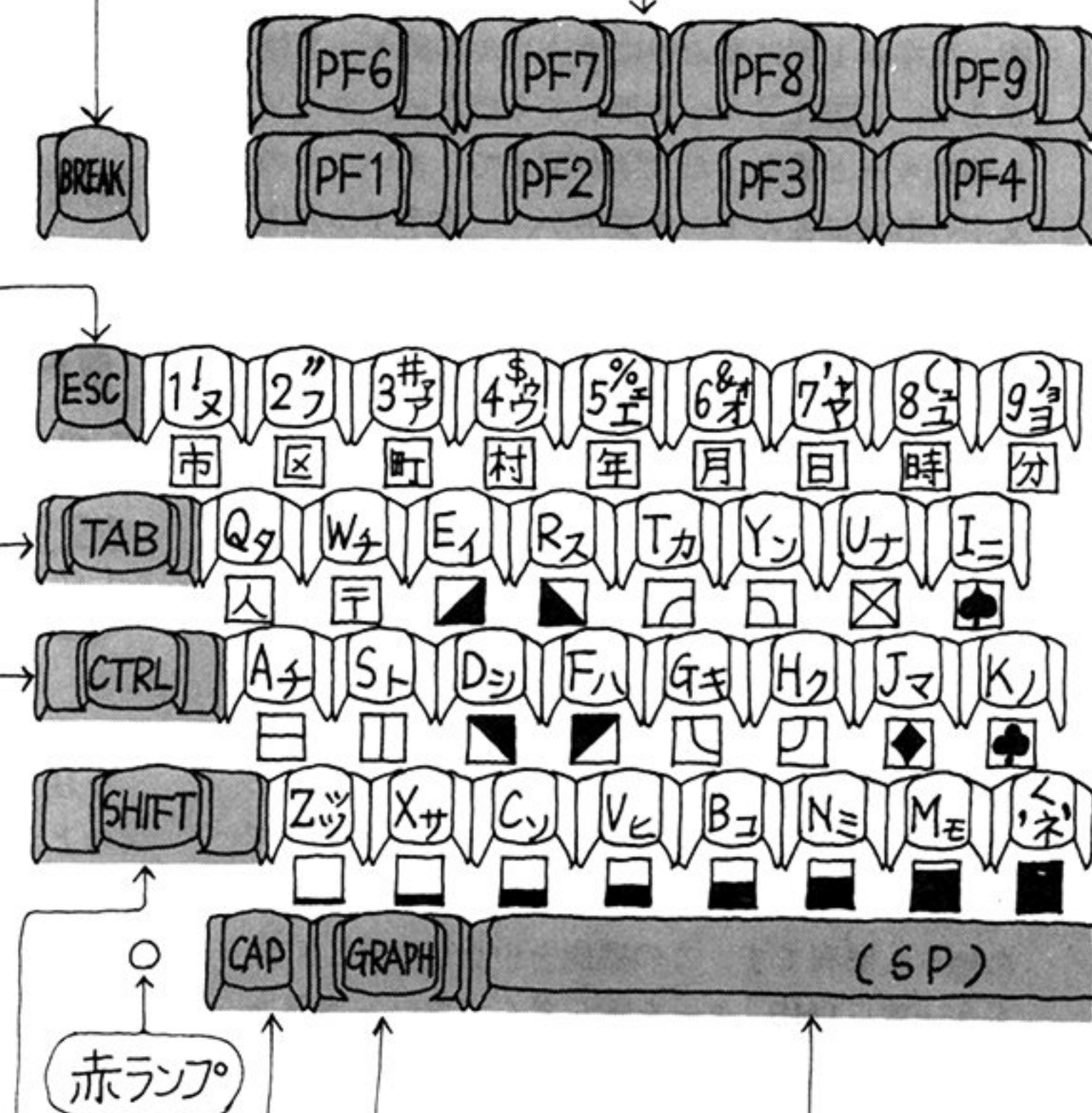
PF（プログラマブル　ファンクション）キー．
PF1～PF10，
コンピュータに対するいろいろな指示を与えるキー。機種によって機能が多少異なる。

ESC（エスケープ）キー．
いろいろな働きをする（詳細略）。たとえば、LIST命令で画面に表示している文字を一時停止する。もう一度押すと次の行から出る。解除するには BREAKキーを押す。

TAB（タブ）キー．
カーソルを次のタブ位置（8けた単位）に移動する。

CTRL（コントロール）キー．
いろいろな働きをする（詳細略）。

赤ランプ

SHIFT（シフト）キー．
左右2ヶ所にある。
たとえば、英大文字のとき、このキーを押しながら文字キーを押すと小文字が出る。また、小文字のときは大文字が出る。その他いろいろな働きをする。

CAP（キャピタル）キー．
左の赤ランプがつき、英文字が大文字になる。もう一度押すと、赤ランプは消えて小文字にもどる。

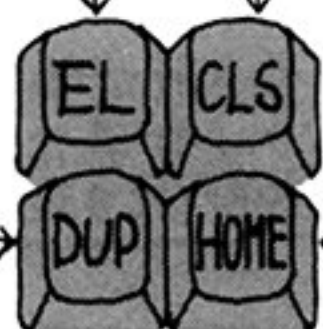
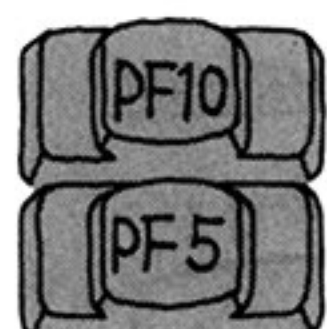
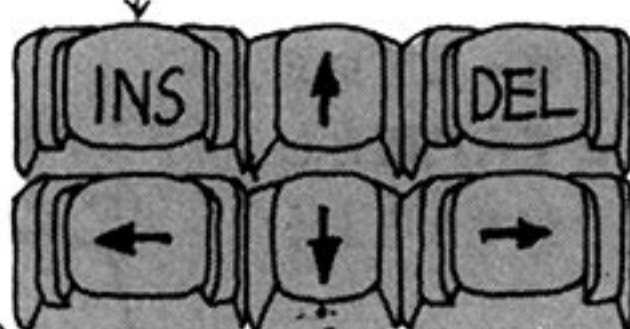
スペースキー．
空白（スペース）が出る。

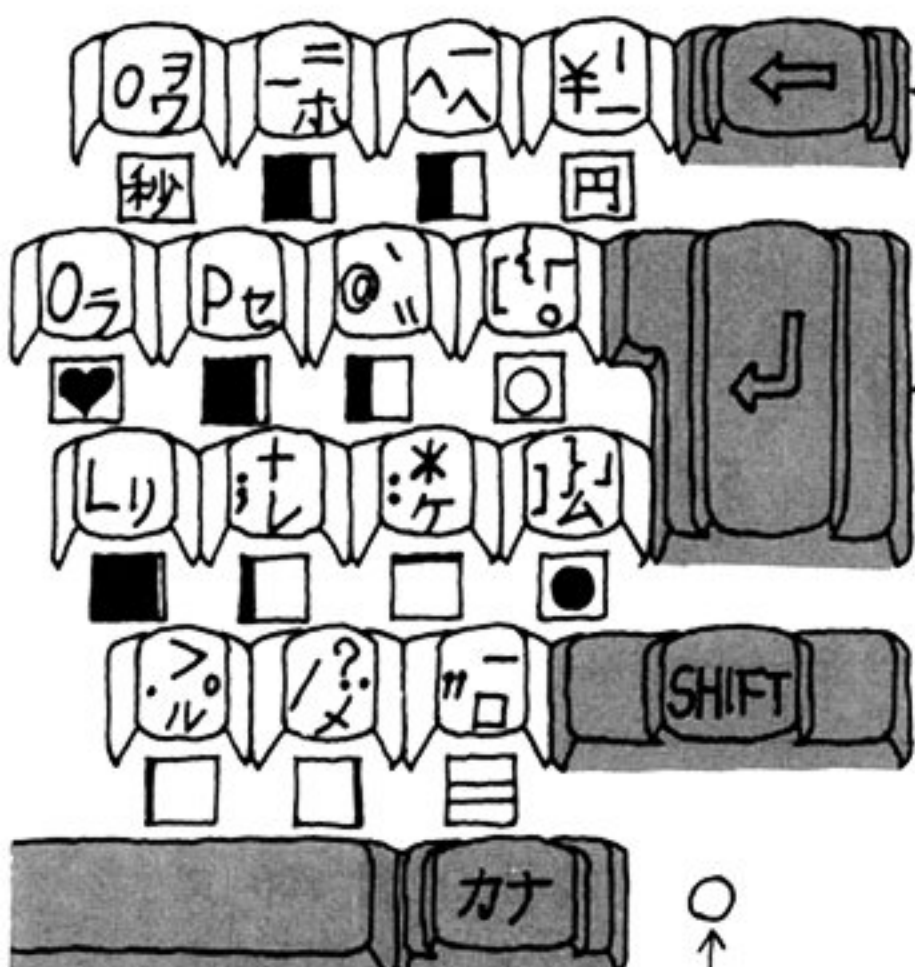
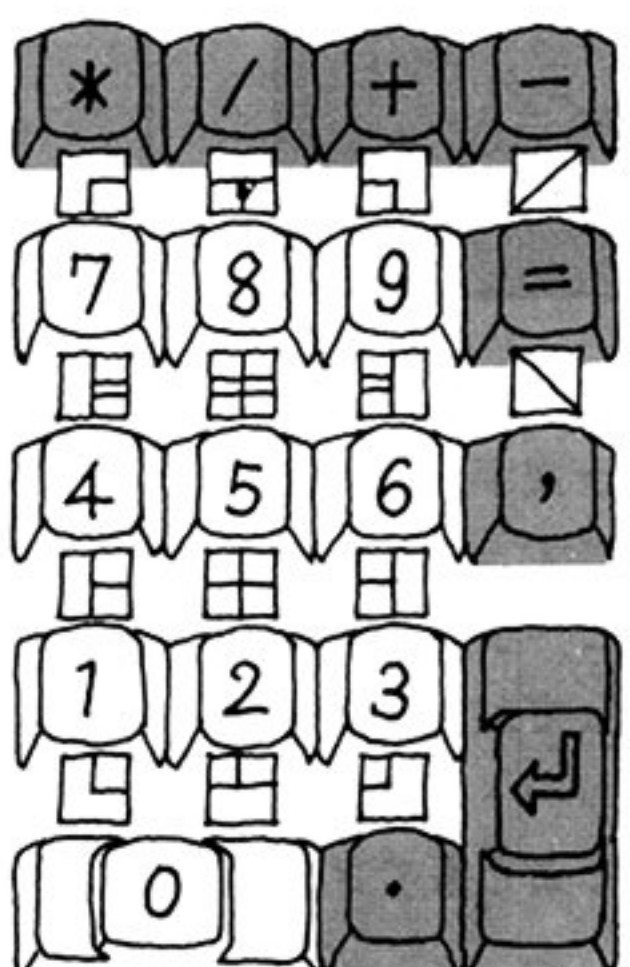
GRAPH（グラフィック）キー．
このキーを押しながら文字キーを押すと上図の文字キーの側面に示すグラフィック記号が出る。

DUP(デュープ)キー.
カーソルの上の行にある文字を 8文字単位で 写しとる.

EL(イレーズライン)キー.
カーソルの右側の文字を全部 消してしまう.

CLS(シーエルエス)キー.
画面の文字が全部 消えて カーソルが ホームポジションに 移動する.
FM-8は CLEAR になっている。

HOME(ホーム)キー.
カーソルが ホームポジションに もどる.

赤ランプ

INS(インサート)キー.
右側の赤ランプがつく. 文字と文字との間に別の文字を入れたいときに使う. もう一度 押すと 消える.

DEL(デリート)キー.
カーソルを文字の上に重ねて このキーを押すと その文字が消える.

赤ランプ

カナキー.
右側の赤ランプがつき、文字キーを押すとカナ文字が出る。もう一度 押すと 赤ランプは消えて 英文字にもどる。

RETURN(リターン)キー.
入力が 終結する.
FM-8は RETURN になっている.

バックスペースキー.
カーソルの左側の文字を 1文字消去する.
FM-8は BS になっている。

カーソル移動キー(矢印のキー).
カーソルを矢印の方に 移動する.

注.) ホームポジションとは CRT画面の 左上偶のこと.

2 BASIC 入門

あなたは，すでにコンピュータを少なくとも電卓がわりに使うか，それ以上のことをやらせることができるようになりました．もちろん，それだけではコンピュータを使えるとはいえません．

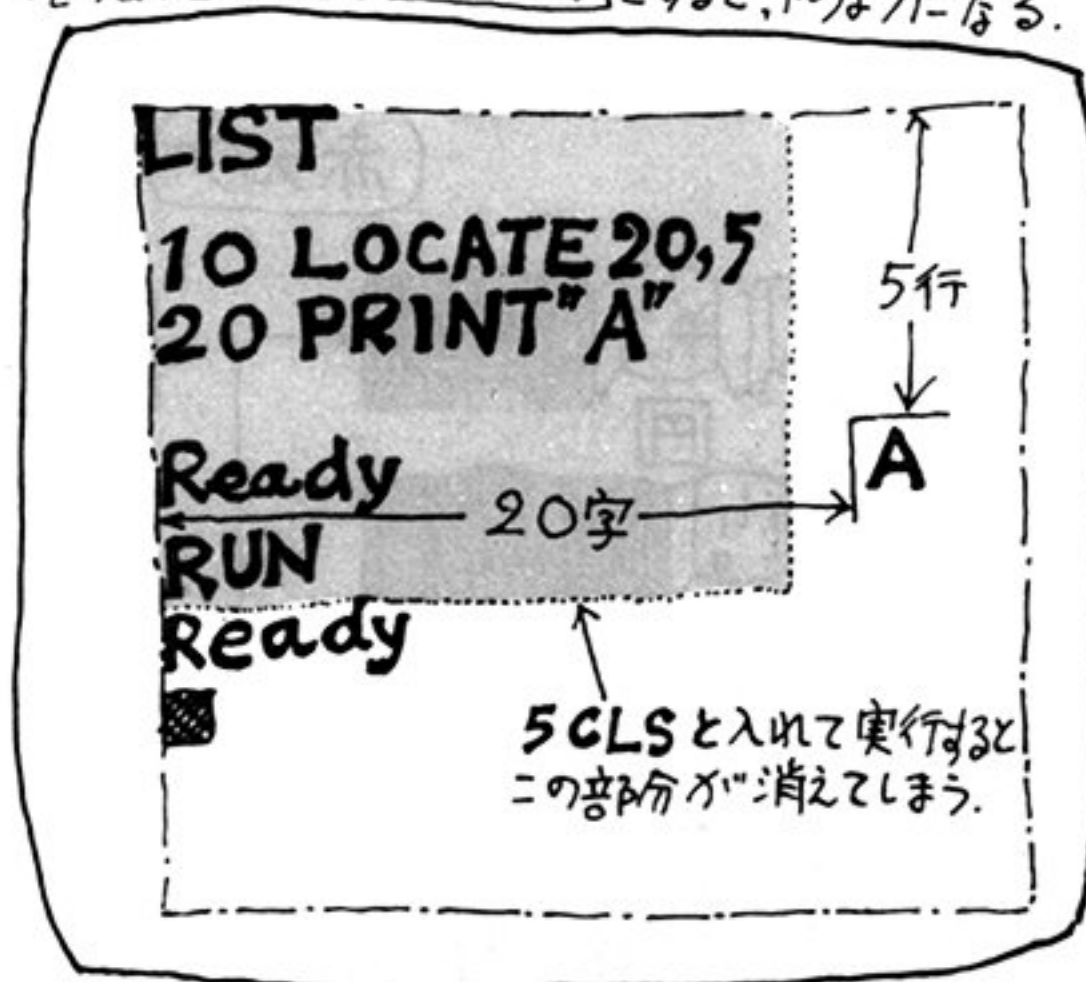
ここからが，いよいよBASICを使ったプログラム作りの本論です．ここでは，まず入門編として，必要最小限度のプログラム作りができるように，話を進めてみることにします．

2.1 文字の表示

F−BASICで利用する文字（文字セット）は，英字（大文字，小文字），数字，カナ文字，特殊記号，そしてグラフィック文字などです．これらには，付表に示すようなキャラクタコード（文字に付けられたコード番号）が付けられており，キーボートからキーインする以外に，コードで指定して表示したり，印刷し出したりすることもできます．

これらの文字や記号は，プログラムを作るときに欠かせないものですが，グラフィック文字で，グラフや図形を描き出してやることも可能です．キーボードから英文字や数字文字記号を表示する方法は，ご存知のとおりですが，カナ文字を表示したいときは，[カナ]キーを押してください．キーの右側に赤ランプがつき，キーを押すと，キー表面右下に書いてある文字が表示されます．また[SHIFT]キーを押しながら押すと，表面右上に文字が書いてあるキーの場合，その文字が表示されます．

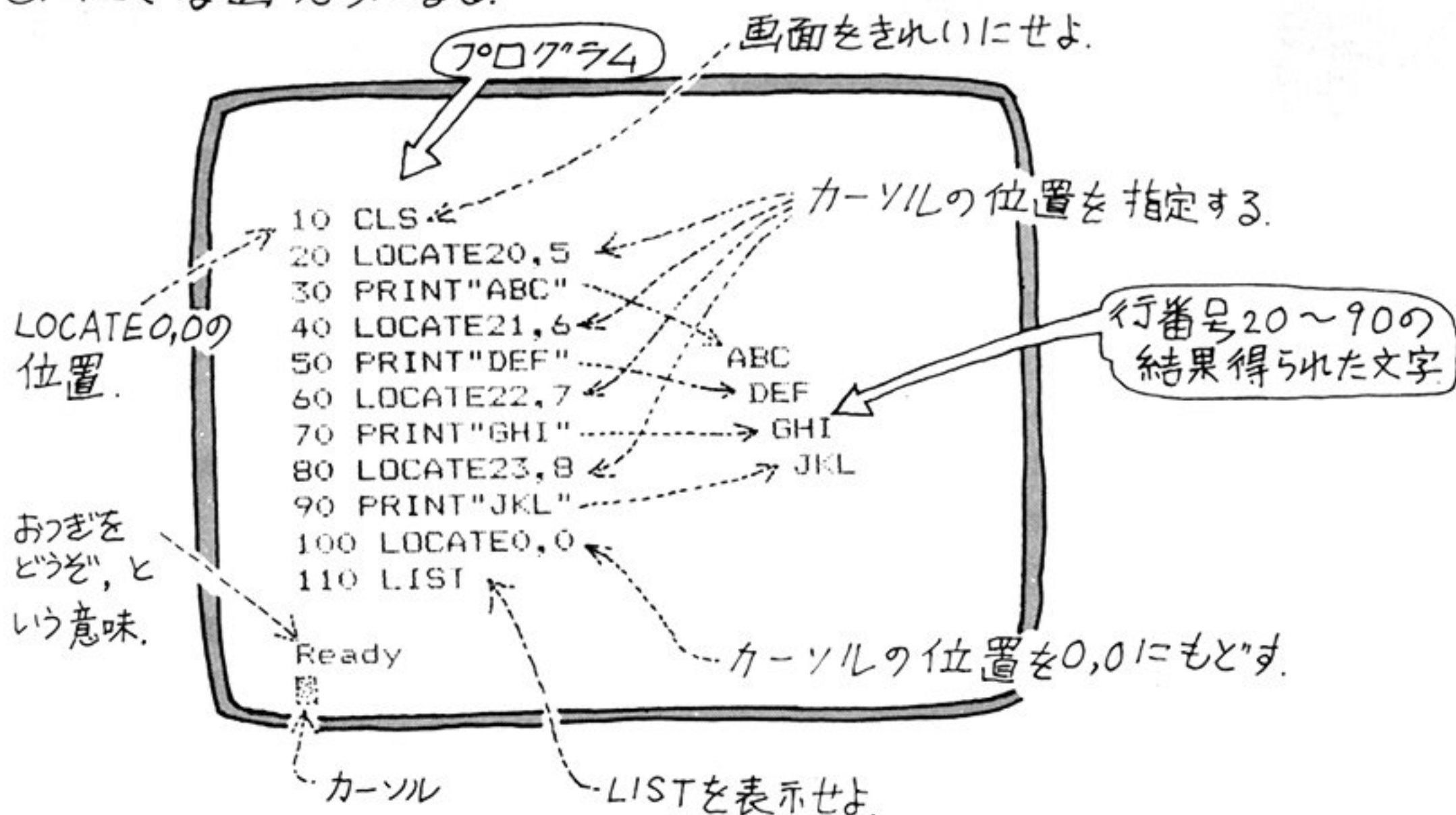
グラフィック文字を使いたいときは，[GRAPH]キーを押しながら目的のキーを押してください．

CRTは初期状態では40×20=800字になっている．
WIDTH 80,25 [RETURN]
と入れると 80×25=2000字になる．
プログラムを入れてから LIST [RETURN] と入れると プログラムが表示される．
そのあと RUN [RETURN] とすると，下のようになる．

◆ CRTは，横40文字，縦20行，合計800文字のます目でできており，任意の位置を座標指定することができます．

例として，横20文字，縦5行おいた次の位置にAという文字を表示してみましょう．プログラムを記憶させてから，一度画面をきれいに（[CLS]キーを押す）したあと，LISTを出してRUNさせてみます．そのときのCRTの状態は，上図のようになっています．

◆ プログラム上で，画面をきれいにしてからAを表示したいときは，CLSという命令を入れてやれば可能になります．たとえば，

5 CLS

というプログラムを追加してRUNすると，上図▒▒の部分が消えた状態で，Aが出ます．

なお，CLSの使い方には，この他にもいろいろな場合があります．とくに，後述のCONSOLEというコマンドと合わせて使うとき，画面の全部を消してしまうだけでなく，その一部を消したりできる方法がありますが，ここでは省略します．詳細は文法書を参照してください．

◆ 画面をきれいにしたついでに，カーソルの表示を消してしまう方法もおぼえておきましょう．それには行番号10のLOCATEという命令（このような命令語をキーワードという）のあとに，0を追加してやればよいのです．

10　LOCATE 20，5，0

これでRUNさせてやると，それまで点滅していたカーソルが消えてしまいます．画面処理をしているとき，カーソルの表面がじゃまになるときがありますが，そんなとき利用する方法です．それにしても，カーソルの表示がないのは困りますね．とにかくカーソルがどこかにあるはずですから，文字は表示されます．

LOCATE 20，5，1 RETURN

と，直接モードで入れ，カーソルを出しておきましょう．この文字が1であれば表示，0なら消えてしまいます．また，これを省略すると1とみなされ，消えたりしません．

◆ 上記のようなプログラムを作るとき，LOCATEなどの長いキーワードをいちいちキーインするのが面倒とお考えの方は，次のキーワードに限り省略形を利用することができます．なお，プログラムの中に省略形を使っても，あとでLISTを表示させてみると，省略しないキーワードに直っているのがわかります．

キーワード	省　略　形
CONT	C.
LOAD	LO.
SAVE	SA.
SKIPF	SK.
MERGE	ME.
GOTO	GO.
GOSUB	GOS.
RETURN	RET.
RANDOMIZE	RNDM.
LOCATE	LOC.
COLOR	COL.
MOTOR	M.
RUN	R.
LIST	L.
CONSOLE	CONS.
WIDTH	W.

2.2 数値定数

これから述べるプログラムの例では，どうしても数値を扱う話が多くなり，申しわけないと思います．コンピュータは，単なる計算機械ではなく，人々の思考活動や創造活動までも，側面からアドバイスできる道具です．そのためには，必ずしも計算ばかりが必要なわけではありません．

しかし，コンピュータが数値で動く機械であるため，プログラムの構造を知っていただく上で，数値や計算式を使う説明が，もっとも簡潔にできますし，また，たとえ事務処理に使う場合でも，欠くことのできないのが数字と計算です．

さらにいいかえると，計算処理をともなう事務処理こそコンピュータにとって，もっとも得意とする分野である，といってもよいのかも知れません．では，早速簡単な計算のプログラムから実行することにします．

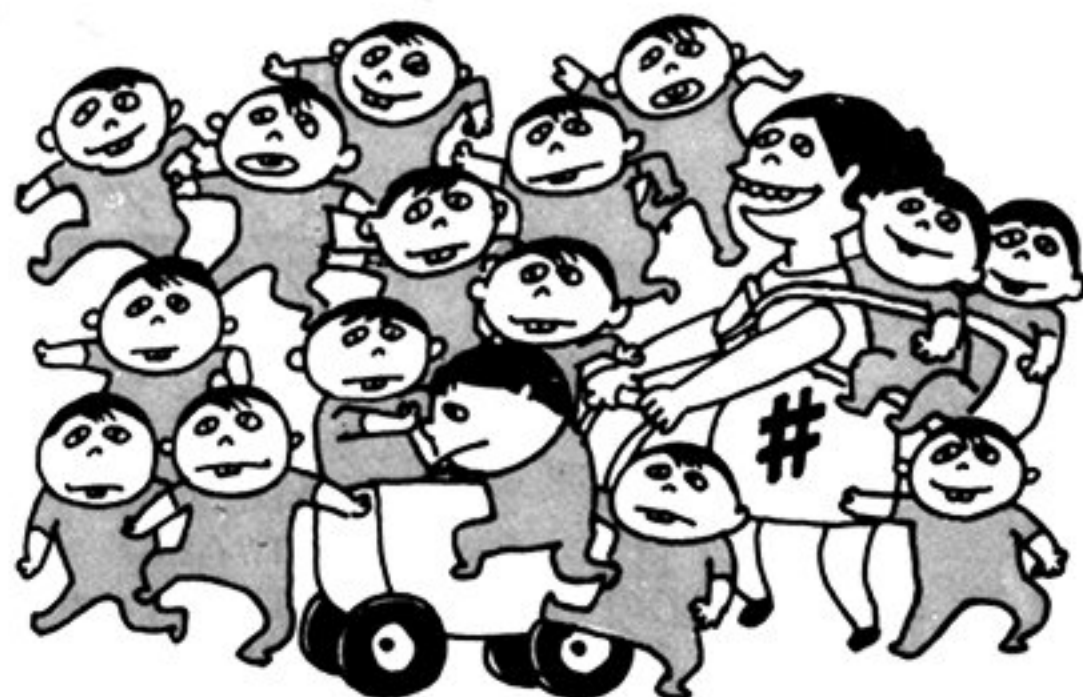
```
10 A=1/3
20 PRINT A
```

行番号10に/（スラッシュ）の記号が使われていますが，これはもちろん割り算の記号です．同様にして，掛け算には＊（アスタリスク）が使われます．1÷3ですから答は0.333……と無限に続くはずです．ところが実行してみると．

```
RUN
 .333333
```

と6桁しか表示されません．コンピュータの中では7桁までの精度で記憶されていますが，表示は6桁なのです．無限に計算させるのは無理だとしても，何百桁も表示させるには，それなりのプログラムを作ってやらないと得ることができません．そのような手続きを取らずに，もっと精度の高い結果を得ることも可能です．この場合，#（シャープ）記号を付けてやります．

```
10 A#=1#/3#
20 PRINT A#
RUN
 .3333333333333333
```

今度は16桁までの精度で表示されました．コンピュータの中では17桁までの精度で記憶されています．

◆ **単精度数値定数と単精度変数**

#マークの付かない初めの数値を単精度数値定数，そして変数を単精度変数といい，コンピュータの中では4バイト分の領域を占領します．

多くの場合，これだけの桁数あれば充分間に合うはずです．もう一度単精度数値定数をまとめてみると次のようになります．

単精度数値定数

(1) 有効数字が7桁以下の定数

(2) Eを用いた指数形式で表現されたもの

私達は数字で大きな数や，小さな数を表わすとき，たとえば 3×10^7 とか 3×10^{-7} などと書きます．コンピュータでは 10^{-7} のような表現がとれないため．

3×10^7 ………3E＋7

3×10^{-7}………3E－7

のような表現をします．

(3) 最後に感嘆符！が書かれた定数

例
```
PRINT1234567*123456    RETURN
 1.52415E+11

PRINT12345670!*12345670!    RETURN
 1.52416E+14
```

この場合！を省略すると，倍精度の演算を実行します．

倍精度数値定数

(1) 有効数字が8桁以上の定数

(2) Dを用いた指数形式で表現されたもの

単精度のEのかわりにDを用います．

3×10^8 ………3D＋8

3×10^{-8}………3D－8

（3） 最後に＃記号の付いた定数

```
例PRINT 12345670              RETURN
   *12345670

 PRINT 12345670*123          RETURN
  152415567748900
```

◆ 単に定数といった場合，大きく分けると文字定数と数値定数の二種類になります．文字定数とは，〃〃でくくった文字列のことです．文字定数については章を改めて，いろいろな性質を調べてみることにしましょう．

数値定数は，厳密にいうと整数形式，固定小数点形式，浮動小数点形式，16進形式，8進形式の五つの形式があります．ここではそれらについて簡単にふれてみましょう．

（1） 整数形式：――＋，－の符号のあと（＋は省略）に整数が並んだ数値で，－32768～＋32767までの範囲です．

（2） 固定小数点形式：――＋，－の符号のあとに小数点を含んだ数値（1.0，－123.4，.95）をいいます．

（3） 浮動小数点形式：――上の例で出てきたようにEやDなどの指数部を含んだ数値（1.5532E＋7，1.000234D－8）をいいます．

（4） 16進形式：――16進数で表わされた数値で，次の例のように，&Hのあとに最大4個まで並べることができます．なおこの形式でインプットされた数値が，アウトプットされるときは自動的に，10進数に直されてしまいます．

例 &H6B，&H003F

```
A=&HB23:PRINT A              RETURN
 2851
```

（5） 8進形式：――同様に8進数の数値で&Oが使われます．この場合は最大6個まで並べることができます．

例 &O 0123，&O 124716

```
10 A=&O1234
20 PRINT A
RUN
 668
```

2.3 算術式

コンピュータでいう式とは，演算の方法を示すもので，定数，変数，関数を演算子で結んで指定します．もちろん定数だけであってもかまいません．変数や関数についてはあとで説明します．

コンピュータで扱う式にはいろいろあります．ここでは，差し当たって必要な算術式をいくつかの例によって説明します．

算術式というのは，変数，定数，関数などの数値データを算術演算子で結び合わせて作ったものです．もちろん，数値データのみの場合も算術式といいます．

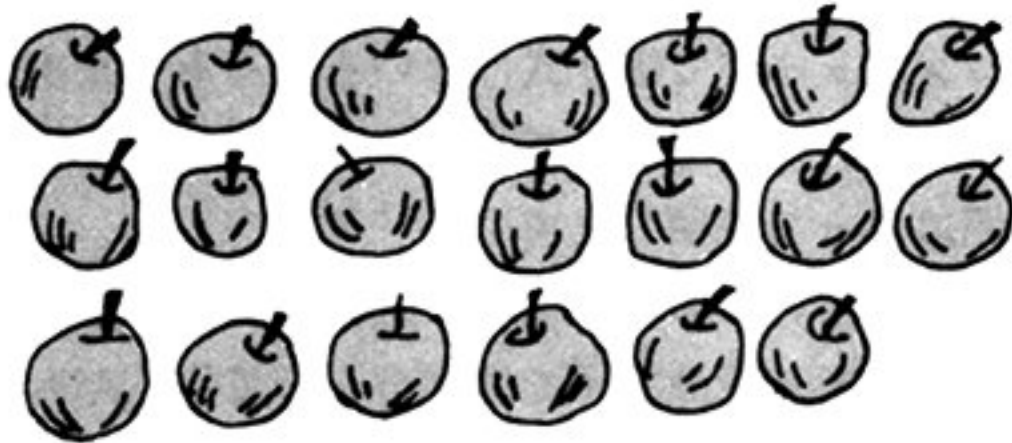
コンピュータでいう算術演算子には，次のようなものが使われます．数学の場合も，たとえば（ ）があるときは，内側のかっこの中が第一優先，次に×と÷，第3番目に＋と－があるように，やはり優先順位があり，基本的には数学の場合と同じように並んでいます．（ ）の中が第一優先になる点も同じです．

優先順位	算術演算子	説　　明
1	∧	べき乗
2	＊	乗　算
	／	浮動小数点除算
3	¥	整数の除算，結果は浮動小数点除算の小数点以下を切り捨てた値
4	MOD	整数の除算の剰余
5	＋	加　算
	－	減　算

なお，¥，MODでは，演算に使われる数字が整数形式でない場合は，小数部分が四捨五入され，整数に丸められた後に，演算が実行されます．

◆ 円の面積を求めてみましょう．

```
10 HANKEI=5
20 MENSEKI=3.14*5^2
30 PRINT MENSEKI
RUN
 78.5
```

コンピュータは，ちゃんと優先順位を守ってくれましたね．ではこんなのはどうでしょう．

◆ 20個のリンゴを3人で分けたら余りは？　つまり，20÷3＝6余り2です．MODを使えば簡単に出るはずです．答は2になるでしょうか？　今度は直接モードでやってみます．

```
PRINT 20MOD3          RETURN
 2
```

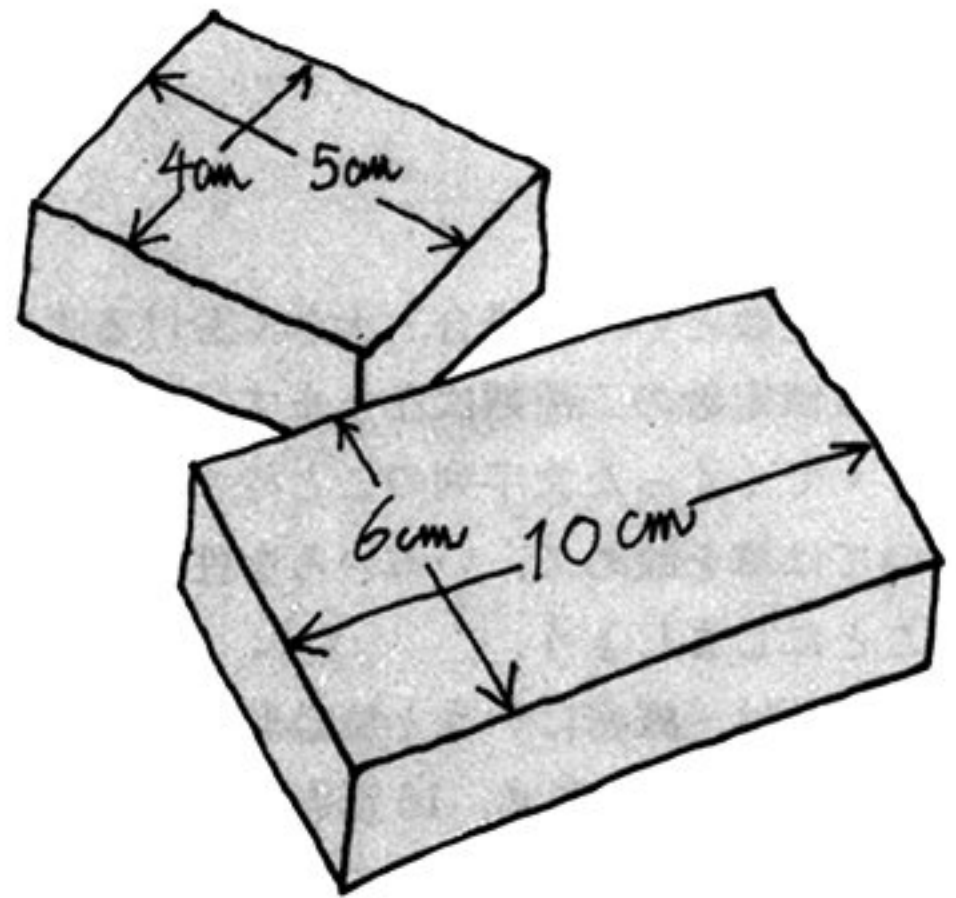
◆ 2÷6は0.333ですね．¥を使って計算すると，四捨五入されて0になるはずです．

```
PRINT 2¥6             RETURN
 0
```

◆ ¥やMODの例をいくつか試してみましょう．

```
PRINT 15.5 MOD 5      RETURN
 1
PRINT 15.4MOD5        RETURN
 0
PRINT 15.5¥5.4        RETURN
 3
PRINT 15.4¥5.5        RETURN
 2
```

簡単ですね．たとえば15.5 MOD 5は，まず15.5が四捨五入されて16になったあと，5で除算が実行され，余りの答えが出てきたわけです．また，15.4¥5.5は，四捨五入した結果15÷6となります．その結果は2.5ですが，小数点以下が切り捨てられるので，結果が2となってしまうわけです．

◆ 図のような二つのようかんを，3人に平方センチ単位できちんと分け，余りをあなたが食べるとしたら？

まず，二つのようかんは4×5＋6×10＝80cm²

そこで3人で分けると　80÷3＝26余り2cm²

あなたの取り分は2cm²．甘い物は，余り食べない方がよいのです．コンピュータでやってみましょう．

```
PRINT 4*5+6*10MOD3    RETURN
 20
```

おや？　変ですね．そうです．優先順位のことをすっかり忘れていました．INS キーを押して(　)をそう入し，もう一度試してみましょう．

```
PRINT(4*5+6*10)MOD3   RETURN
 2
```

今度はうまくいきました．(　)は幾つ使ってもかまいません．数学の場合は〔｛(　)｝〕と使い分けますが，コンピュータでは，全て(　)でよいのです．ただし，右向きのかっこの数と左向きのかっこの数とは，必ず同じでなくてはなりません．

```
PRINT(10+5*(3+2)/7¥6+1)
 11
```

この答えが正しいかどうか，紙に書いて計算してみてください．それによって，この演算の少し込み入ったルールにもなれていただけるでしょう．

かっこの使い方で，数学の場合，a×(b＋c)をa(b＋c)と書いても，いっこうに差しつかえありません．コンピュータの場合は，a＊(b＋c)と，必ず算術演算子でつないでやらないと，エラーになります．習慣上，つい忘れてしまいがちです．

2.4 ファンクションキー (FM-8の場合)

キーボード上部に並んでいるPF1～PF10，つまりプログラマブル・ファンクションキーの使い方を紹介します．今まで，たとえばPRINTなどの文字がたくさん出てきましたが，いちいちキーインするのが面倒なとき，このキーを利用しようというわけです．ファンクションキーは，初期の状態では，次のような文字列が設定されています．なお，Cr マーク（キャリッジ・リターン）は，RETURN キーを押したのと同じ働きをすることを意味します．また，␣はスペース（空白）を意味します．

PF1	AUTO␣
PF2	LIST Cr
PF3	RUN Cr
PF4	GO␣TO␣
PF5	CONT Cr
PF6	?DATE$, TIME$ Cr
PF7	LIST "LPT0:" Cr
PF8	KEY␣
PF9	LOAD␣
PF10	HARDC Cr

ここに示したものはFM－8の場合の例です．FM－7，FM－11については次ページで示します．もう，何回も出てきたコマンドもありますね．これ以外のコマンドについては，順次説明します．

プログラマブル・ファンクションキーを押したとき，別のコマンドが出るようにしたいときもありますね．このようなときに必要になるのが，次の命令です．

KEY　ファンクションキー番号，文字列

文字列の部分は最大15文字まで，または＋記号でCHR$関数（後述）をつないだ形で指定されます．例を見てみましょう．

◆ KEY 10, "PRINT" RETURN

これで，PF10のキーを押すと，CRT上に，PRINTと表示されるようになります．ここで文字列は" "でくくった文字の並びか，付録に示すキャラクタコード表から作ることができます．そこでキャリッジリターンも同時に実行させたいときは，キャラクタコード表の Cr のコード，すなわち十六進数のD，十進数に直すと13から，次のようにして作ることができます．なお，CHR$というコマンドの詳細は後述します．

◆ KEY 9, "KEY LIST" +CHR$ (13) RETURN

ここで KEY LIST というのは，プログラマブル・ファンクションキーの内容を表示せよ，という命令です．そして，CHR$ (13) によって RETURN キーを押したのと同じ動作をするようになります．では，これらの命令を入れた後，実際にPF9を押してみましょう．FM－8では，つぎのようなPFキーの一覧表が表示されてくるはずです．この命令自身は，もちろんFMシリーズのどの機種にも共通に適用できます．

PF1	AUTO␣
PF2	LIST Cr
PF3	RUN Cr
PF4	GO␣TO␣
PF5	CONT Cr
PF6	?DATE$, TIME$ Cr
PF7	LIST "LPT0:" Cr
PF8	KEY␣
PF9	KEY LIST Cr
PF10	PRINT

◆ ROMモードとDISKモード

FM－7，FM－8のパソコンをBASICで働かせる場合，いずれも2種類のBASICが用意されています．

その一つは，コンピュータに内蔵されたLSI（ROM）の中に記憶しているものです．したがって，フロッピィディスクが動作しない状態でコンピュータに電源を入れると，ただちにこのBASICが動作するようになります．そこで，このBASICのことをROMモードといいます．

もう一つは，DISKモードといい，外部記憶装置としてフロッピィディスク装置を接続した状態で利用する場合です．ROMモードのBASICではフロッピィディスク装置を利用することはできません．フロッピィディスクも利用できるBASICはDISKモードといい，これはフロッピィディスクの中に記憶されています．そこで，DISKモードのBASICの入っているフロッピィをディスク装置の0と書いてある所に入れ，コンピュータの電源を入れるかリセットボタンを押すと，ROMモードからDISKモードに置き換えられます．

ROMモードとDISKモードの違いは，フロッピィディスク装置を利用できるかどうかという点だけです．なお，FM－11は原則としてDISKモードしかありません．

◆ FM－7のPF

FM－7のプログラマブル・ファンクションキーは，ROMモードの場合とDISKモードの場合とは多少違いがあるので，表として示しておきましょう．いうまでもなく，これは初期設定の状態（電源を入れたときの状態）で，前述のようにKEY命令を使うと変わってきます．

モード／PFキー	ROMモード	DISKモード
PF1	AUTO␣	AUTO␣
PF2	LIST CR	LIST CR
PF3	RUN CR	RUN CR
PF4	CONT CR	CONT CR
PF5	LLIST CR	LLIST CR
PF6	LOAD CR	LOAD"
PF7	SAVE"	SAVE"
PF8	?DATE$, TIME$	FILES
PF9	SCREEN 7, 7 CR	SCREEN 7, 7 CR
PF10	HARDC CR	HARDC CR

左表に示すコマンドの中に，LLISTというのがありますが，これはコンピュータの中に記憶しているプログラムをプリンタで印刷するための命令です．

LLIST [RETURN]

と入力すると，接続してあるプリンタ装置がプログラムの全部を印刷します．

LLIST200 [RETURN]

行番号200のプログラム

LLIST30－200 または

LLIST30，200 [RETURN]

行番号30から200までのプログラム

LLIST－50 または

LLIST，50 [RETURN]

行番号50までのプログラム

LLIST300－ または

LLIST300，[RETURN]

行番号300以降のプログラム

ただしこの命令は，初めにCRTに表示される文字が，Version 1.0 になっているFM－8の場合は使えません．この場合には，LIST″LPT 0:″という命令を利用してください．なお，これはFM－7やFM－11でも利用できます．

LIST″ LPT 0:″ [RETURN]

プログラムの全部を印刷．

LIST″ LPT 0:″ 300，500

行番号300から500までのプログラムを印刷

◆ FM－11のPF

FM－11は，その中にEX，AD，STという三種類の機種があり，そのプログラマブル・ファンクションキーは，次のようになっています．

PFキー ＼ タイプ	FM－11 EX，AD	FM－11 ST
PF1	AUTO␣	AUTO␣
PF2	LIST CR	LIST CR
PF3	RUN CR	RUN CR
PF4	CONT CR	CONT CR
PF5	LLIST CR	LLIST CR
PF6	LOAD″	LOAD CR
PF7	SAVE″	SAVE″
PF8	FILES	?DATE$，TIME$ CR
PF9	SCREEN	SCREEN
PF10	HARDC CR	HARDC CR

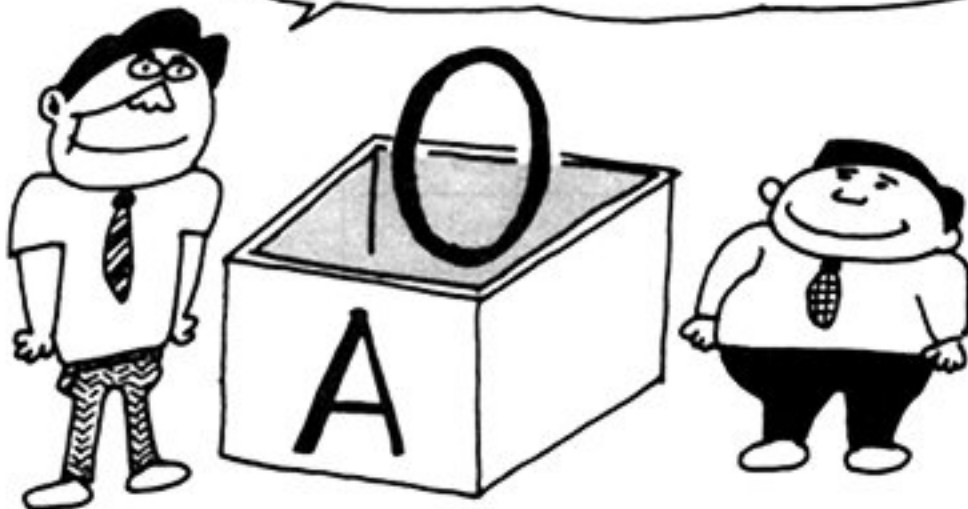
2.5 変　数

もう変数の話はすでに出てきました．数学でいう変数とは違い，定数を入れておく箱，そしてその箱に付けられた名前が変数でした．また，いくつ箱が置いてあっても，別に驚くことはないし，変数に数字（または文字）を入れてやらない限り，その中味は0であることもすでに出てきました．

変数には，数値を入れておく数値変数と，文字列を入れておく文字変数があります．数値変数の方は，もう何度も出てきましたね．文字変数は，いろいろな種類の文字変数を用意しておき，コンピュータ処理の結果によって，その配列や順序を変えてCRTに表示してやるなど，いろいろと楽しい使い方が可能です．その使い分けについては，後章にゆずります．楽しみにしておいてください．

変数に付ける名前は，英数字のいずれでもよく，最大255字までです．ただし，第1文字目は英文字でなくてはなりません．また，英大文字と英小文字で区別することはできません．さらに，BASICに使われる予約語（ステートメント名，コマンド名，関数名，演算子名など）を使うことは許されません．ステートメントなのか変数名なのか，コンピュータには区別できないからです．

変数を準備しただけの状態のときは0が入ってるよ

A

```
10 PRINT"A=";A
20 A=5
30 PRINT"A=";A
RUN
A= 0
A= 5
```

いきなりAの値を表示させる．

A=5としてからAの値を表示させる．

Aに5を入れよ

A

もちろん英文字1字であってもかまいません．ただ，たくさんの種類の変数が出てくる場合，プログラムを作っていて，何にどの変数名を使ったか，ついわからなくなってしまうこともあります．そこで，キーインが面倒にならない程度の長さの，わかりやすい変数名にしておくと便利です．

変数名は，255文字以内なら長さには制限ありませんが，コンピュータが違う変数として区別できる長さは，16字以内に限られます．16字まで同じで，17字以後が違っていても，同じ変数名と見なされてしまうわけです．

変数のあとに記号を付けて，変数の型を示す方法（型宣言）は，すでに数値定数のところで出てきました．ここで新しく二つの種類を追加して，変数の型宣言文字をあげてみます．

%	整数変数を示す． 一つの変数につき，2バイトのデータ格納域を必要とする．
!	単精度変数を示す． 一つの変数につき，4バイトのデータ格納域を必要とする．
#	倍精度変数を示す． 一つの変数につき，8バイトのデータ格納域を必要とする．
$	文字変数を示す． 最大255文字のデータを格納することができる．

例によって，簡単な実例でその内容を調べてみましょう．

```
10 A%=9.876
20 B%=2.345
30 PRINT A%+B%
RUN
 12
```

A％は整数変数であるため，右辺の数が四捨五入されて10となり，B％が2となって記憶され，その結果が12となって表示されたのです．

```
10 A=1.234567+1.020304
20 PRINT A
RUN
 2.25487
```

変数Aは，数値を入れておく箱です．3＋5というような式は入りません．そこで行番号10では，Aに演算結果が入っているわけですね．なお，型宣言をともなわない変数名は，このように単精度形式の数値変数として扱われます．

```
10 A#=123456789
20 B#=876543210
30 PRINT A#+B#
RUN
 999999999
```

今度は倍精度変数として扱われています．

```
10 A$="ｱｲｳｴｵ"
20 B$="ｶｷｸｹｺ"
30 PRINT A$+B$
RUN
ｱｲｳｴｵｶｷｸｹｺ
```

```
10 A$="ﾔﾚ ｳﾂﾅ "
20 A$=A$+"ﾊｴ ｶﾞ ﾃ ｦ ｽﾙ "
30 A$=A$+"ｱｼ ｦ ｽﾙ"
40 PRINT A$
RUN
ﾔﾚ ｳﾂﾅ ﾊｴ ｶﾞ ﾃﾞ ｦ ｽﾙ ｱｼ ｦ ｽﾙ
```

文字列の取り扱いについては，この辺でやめ，あとの楽しみにしておくことにしましょう．

◆ DEF/INT/SNG/DBL/STR

変数にはいろいろな型があり，いちいち%，！，#，$などの記号を付けて宣言しなくてはなりません．しかしときには，このような手続きが面倒になってくる場合もあります．そんなとき，まとめてこの文字は何型だと指定してしまう方法がDEF～です．DEFはdefine（定義する）の略ですね．

DEF型指定　文字の範囲

型指定は，ＩＮＴ（整数），ＳＮＧ（単精度），DBL（倍精度），STR（文字）のいずれかです．また文字の種類は，アルファベット一文字でなくてはなりません．何文字もDEFで宣言するときは，アルファベットの上位の一文字と，下位の一文字を－で結んでやります．

この宣言によって，宣言された文字の範囲が型指定で示す変数型になってしまいます．ただし，前述のような，型宣言文字を持つ変数は，DEF宣言されている文字より優先して，型宣言された型の変数になってしまいます．

なお，このような宣言が全くない変数は，全て単精度変数として扱われます．

```
10 DEFSTR A-Z
20 X="ｱｲｳｴｵ"
30 A%=10:B%=30
40 X%=A%+B%
50 PRINT X
60 PRINT X%
```

型指定

型宣言の変数

```
RUN
ｱｲｳｴｵ
 40
```

このプログラムによると，行番号10の宣言文で，アルファベットの全ての文字が文字変数として扱われることになったわけですから，変数Xももちろん文字変数です．したがって，RUNさせると，アイウエオとCRT上に表示してきます．もし，行番号10がないと，Xは単精度の数値変数ですから，アイウエオという文字列はXの中に入れることができず，エラーを表示してきます．

使える文字はアルファベット26文字に限られますが，たとえば名簿を作ったりするときなど，いろいろと使い道のある方法です．

2.6 INPUT

コンピュータとあなたとの会話のための，ステートメントがINPUT文です．INPUTのステートメントに出会うと，コンピュータは？という表示をして，あなたの指示を待つ状態になります．そして，あなたがキーボードから必要な指示を与えてやると，それによって再び動作を開始します．

> **INPUT［″プロンプトメッセージ″{;|,}］変数名［,変数名］……**

［　］でくくった部分は，なければなくてもよい，という意味です．また，プロンプトメッセージというのは，コンピュータが何を質問しているのかをわかりやすくするための文字列です．実例でいろいろ調べてみましょう．

美しい湖水．どこまでも広々と続いている湖水の表面．私達には，遠い対岸のかなたまで真っ平らに見えますが，実は真ん中がぐっと盛り上がっていることをご存知ですか？　本当です．その証拠に，広い湖水では水の盛り上がりのため，対岸が見えなくなってしまいます．そうです．地球が丸いからです．

図を見てください．**三平方の定理**（ピタゴラスの定理）から，

$$HANKEI^2=\left(\frac{HABA}{2}\right)^2+(HANKEI-TAKASA)^2$$

これを解くと，

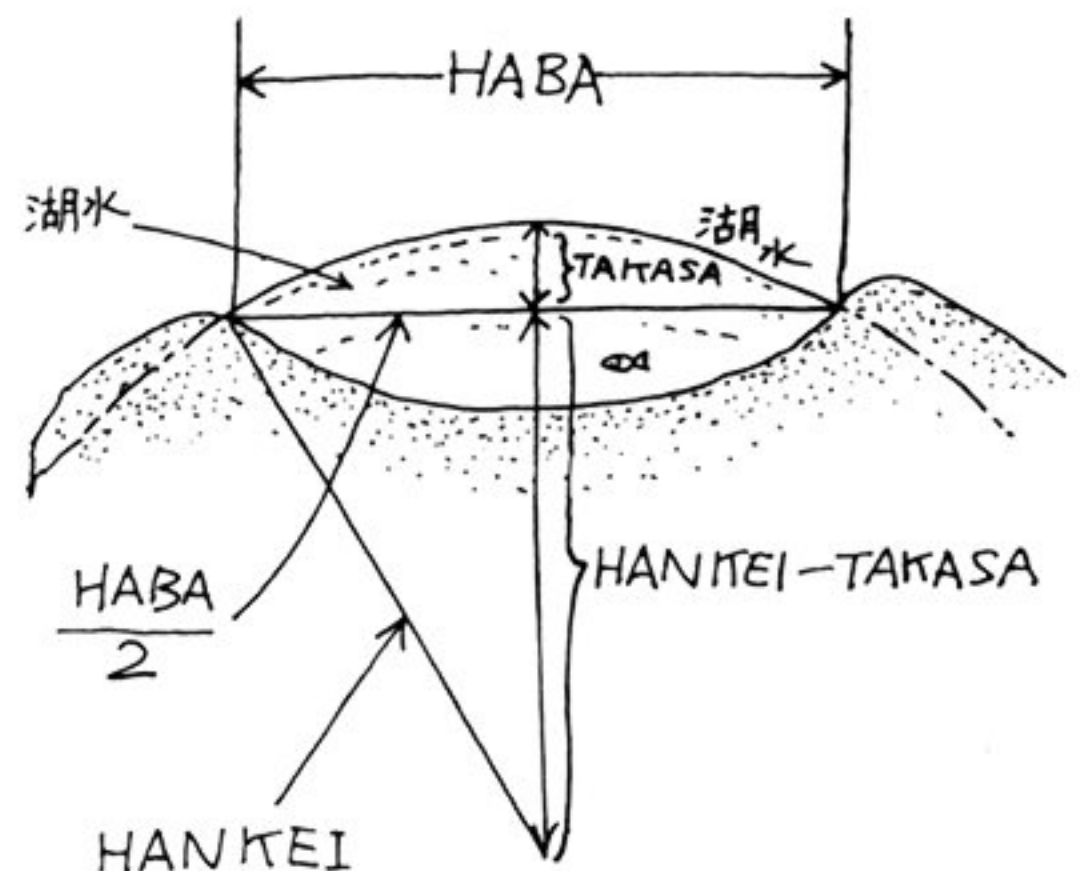
$$HANKEI^2=\frac{HABA^2}{4}+HANKEI^2-2\times HANKEI\times TAKASA-TAKASA^2$$

両辺から $HANKEI^2$ を引くと，

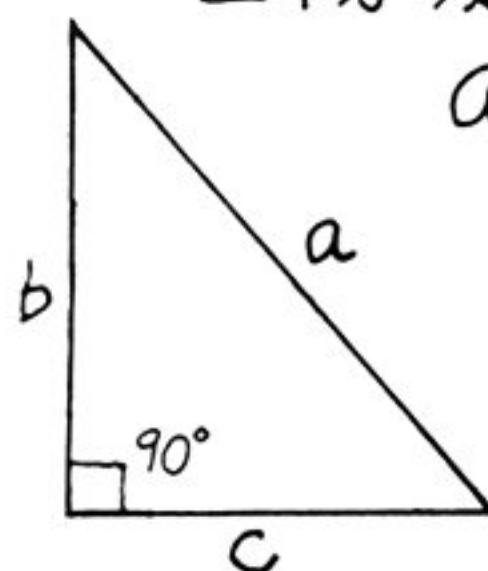
$$0=\frac{HABA^2}{4}-2\times HANKEI\times TAKASA+TAKASA^2$$

他に比べて $TAKASA^2$ はとても小さいので無視し，上式を整理すると，次のようになります．

$$TAKASA=HABA^2/(8\times HANKEI)$$

湖水の水は盛り上がっている

$$HANKEI^2=\left(\frac{HABA}{2}\right)^2+(HANKEI-TAKASA)^2$$

これを解くと

$$TAKASA\fallingdotseq\frac{HABA^2}{8\times HANKEI}$$

ここまでのアルゴリズムはあなたの仕事，あとはコンピュータにまかせればよいのです．

◆ TAKASAを求めるプログラムを作ってみましょう．

```
10 INPUT"ﾊﾝｹｲ ﾄ ﾊﾊﾞ ｦ
ｵｼｴﾃ ｸﾀﾞｻｲ ";HANKEI,HABA
20 TAKASA=HABA^2/(8
*HANKEI)
30 PRINT TAKASA
RUN
ﾊﾝｹｲ ﾄ ﾊﾊﾞ ｦ ｵｼｴﾃ ｸﾀﾞｻｲ ?
```

と表示が出ます．そこで地球の半径6378000（m）と幅4000（m）をコンマで区切って入れてやると，

.313578

と盛り上がりの高さが出てきます．幅4kmの地球上の湖の水は，約30cmの高さで，丸く盛り上がっているわけです．幅10kmならどうなるか，実際に調べてみてください．

キーインするとき6378000と，1個だけ入れて RETURN キーを押すと，?Redo From Start と画面に現われ，再度キーイン待ちになります．このメッセージは，キーインしたデータの数と，変数の数が一致しなかったり，変数の型が違ったりしたときに表示されるものです．

また，文字列のあとの；を，に変えてやると？の表示はなくなってしまいます．なお，プロンプトメッセージは，省略することもできます．

◆ INPUT文は，数値変数ではなく，文字変数であってもかまいません．ただし，変数名のあとに＄印を付けることを忘れないでください．

```
10 INPUT"アナタ ハ ワタシ ヲ
<スキ> ? <キライ> ?",A$
20 PRINT"ワタシ モ アナタ ヲ"
+A$+"デ ス。"
```

あなたが好きなら私も好き，あなたがきらうなら私も……．コンピュータは正直ですね．このように，文字数列を＋でつなげてやることもできるわけです．ここでプロンプトメッセージの後に，？を付けたのは，プロンプトメッセージが入ると，？が表示されないことになっているからです．なお，キーインは，″″でくくってやらなくてもよいのですが，スペースや，；などがある場合には必要です．

```
10 INPUT A,B,C
20 PRINT A+B+C
RUN
? 3,4,5
 12
```

← INPUTする変数が三つある場合は，，で区切って三つ並べる．

← 結果

INPUT文は，今までのプログラム例と違い，途中であなたのキーインを待つ，という状態になるので一見止まってしまったかに見えます．

このような状態のとき，プログラムを中止するには，BREAK キーを押すか，CTRL キーを押した状態で，XまたはCのキーを押してやれば可能です．次にCONTというコマンドをキーインしてやると，INPUT文の初めから実行を再開します．

なお，文字列を入力するときは数字や文字をキーインせず RETURN キーを押すと，空文字列が1回だけ入ったのと同じ結果が実行されます．

◆ LINE INPUT

LINE INPUT［″プロンプトメッセージ″｛；｝］ 文字変数

文字変数に，1行全体の文字列（255文字以内）をそっくり読み込みます．この場合，プロンプトメッセージがあってもなくても？の表示は出ません．また，スペースや，；などがどこにあっても，また″があってもそのままの形で，文字変数に取り込んでくれます．

```
10 LINE INPUT A$
20 PRINT A$
RUN
,;"ABCD"
,;"ABCD"
```

,;"ABCD" ←キーインした文字列

,;"ABCD" ←A＄の結果をコンピュータがCRT上に写し出した状態です．

2.7 GOTO

GOTO ～ は，文字どおり，～のところに飛んで行けという意味です．

GOTO　　行番号

プログラムは，原則として行番号の順序に実行していきますが，このステートメントに出合うと，指定された行番号のところに，飛んで行ってしまうという便利な命令です．いろいろ利用できますね．ただし，長いプログラムを進めるとき，むやみやたらに GOTO 文を多くすると，次第にどこに飛んで行ったかわけがわからなくなり，プログラム上で一種の迷路ができてしまいます．その点さえ注意すれば，たいへん便利な命令です．

◆ 簡単なプログラムで，使い方を調べてみましょう．

```
10 CLS:INPUT"ハンケイ ト ハバ ヲ
オシエテ クダサイ ";A,W
20 H=W^2/(8*A)
30 PRINT H
40 INPUT"ワカッタラ スキナ キー ヲ
オシテ クダサイ";I$
50 GOTO 10
```

すでに出てきた同じプログラムを少し修正しただけです．行番号10の CLS は，画面をきれいに消してしまう命令です．また40は，変数 I ＄に文字を入れよという命令です．何を入れても動き出します．こうしておかないと，プログラムがすぐ次に進んでしまい，答えを見るひまもありません（CLS を使わなければその必要はありません）．そして，行番号50で再び初めに帰り，同じ動作を繰り返します．

この動作を中止したいときは，BREAK キーを押すか，CTRL キーを押した状態で，X または C キーを押してください．そして，再び実行したいときは，CONT RETURN と入れると，先ほど中止した INPUT 文のところから実行を始めます．

```
10 PRINT"A";:GOTO 10
```

さあ，たいへん．A を次から次へと表示し始め，止まらなくなってしまいました．でも，もうおわかりですね．

2.8 ON～GOTO

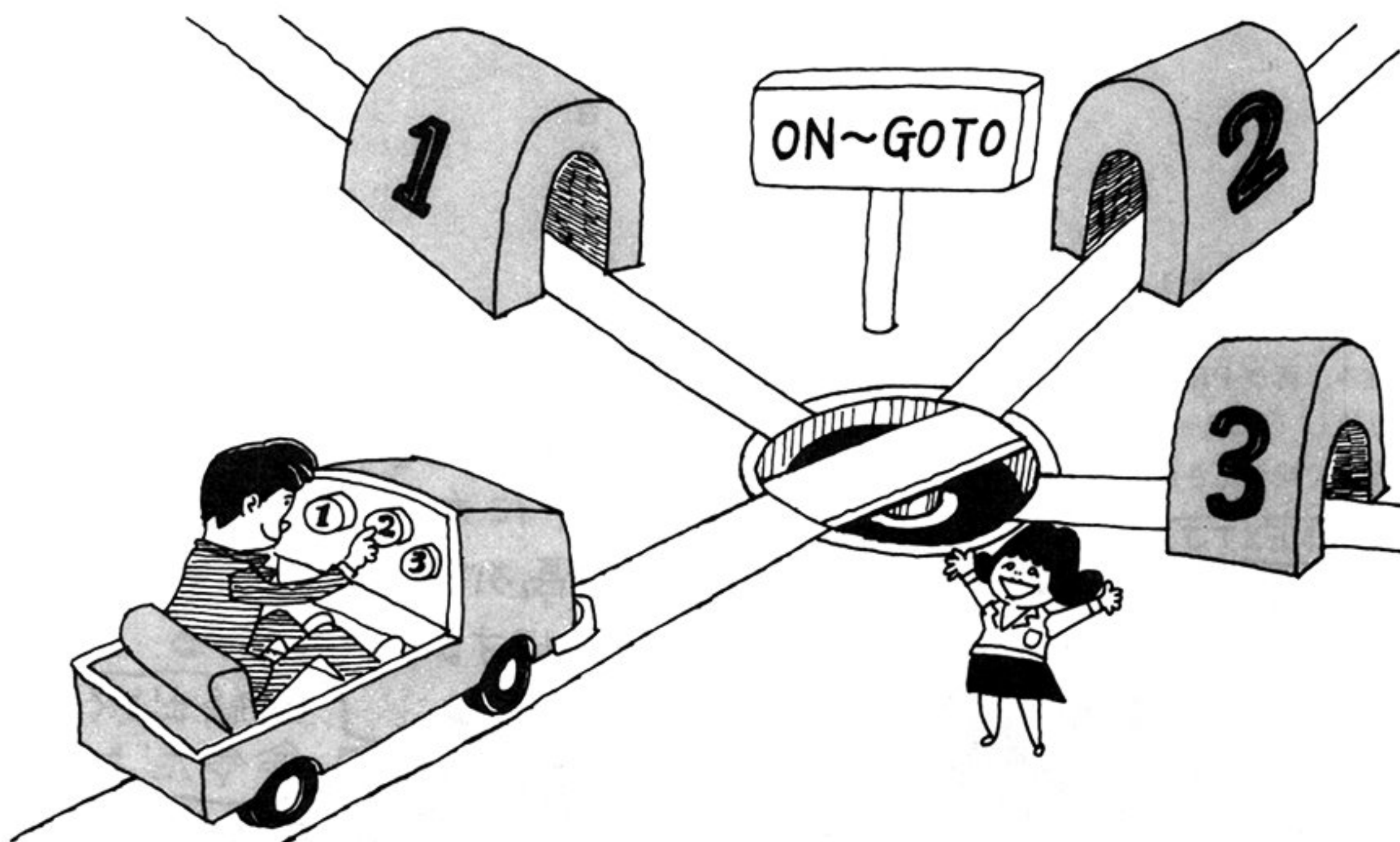
同じGOTOにも，こんな使い方もあります．

ON 式 GOTO 行番号[,行番号]……

式の値が1ならば，GOTOに続く行番号の1番目に，2なら2番目の行番号に……という風に，飛んで行く行番号をいろいろ変えることのできる命令です．ただし，行番号が四つしか並んでいないとき，式の結果が5であれば，このステートメントの部分は飛ばされてしまいます．また，式が整数でない場合には，小数部分は四捨五入されます．

```
10 INPUT"ワカリマシタカ ? ワカッタラ <1>,
   ワカラナイ ナラ <2> ヲ イレテ クダ゛サイ",I
20 ON I GOTO 100,110
30 GOTO 10
100 PRINT"エライ !":GOTO 10
110 PRINT"ザ゛ンネン !":GOTO 10
```

このプログラムによると，コンピュータはあなたにワカリマシタカ？ と，しつこく何度も聞いてくるはずです．うるさくなったら，停止してしまう方法は，もうご存知ですね．なお，Iは単なる変数ではなく，複雑な式であってもかまいません．とにかく結果として1，2……の整数が得られればよいわけです．またGOTO文で飛んで行く先も，そのあとに複雑な処理が待っていてもかまいません．

```
10 CLS
20 PRINT "スウシ゛ ヲ ダ゛スーーー1"
30 PRINT "カナモシ゛ ヲ ダ゛スーー2"
40 PRINT "ローマシ゛ ヲ ダ゛スーー3"
50 PRINT"1 カラ 3 マデ゛ノ"
60 PRINT "スウシ゛ ヲ イレテクダ゛サイ"
70 INPUT I
80 ON I GOTO 100,200,300
100 PRINT "123456":GOTO 1000
200 PRINT "イロハニホヘ":GOTO 1000
300 PRINT "ABCDEF":GOTO 1000
1000 FOR J=1 TO 3000:NEXT J
1010 GOTO 10
```

2.9 FOR〜NEXT

いよいよコンピュータらしい仕事になってきました．同じ種類の仕事を繰り返し実行させるのに便利な命令がこれです．

FOR 変数＝式1　TO　式2［STEP 式3］

NEXT［変数名［, 変数名］……］

繰り返し実行させる命令は，この二つの命令の間に入れてやります．

```
10 FOR J=1 TO 10
20 PRINT "A"
30 NEXT J

Ready
RUN
A
A
A
A
A
A
A
A
A
A
```

Aが10個並ぶ．

つまり，Jが1の状態でNEXT　Jのところまで進み，また初めに帰りJ＝2の状態でまた次に進み，再びPRINT ″A″を実行します．この繰り返し（FOR〜NEXTループ）が，J＝10になるまで繰り返されるわけです．

```
10 FOR J=1 TO 10
20 PRINT "A";J
30 NEXT J

Ready
RUN
A 1
A 2
A 3
A 4
A 5
A 6
A 7
A 8
A 9
A 10
```

同時に変数Jの値を出させると，このようにJが1ずつふえているのがわかります．

Jは1ずつ数が増加していくようすがわかりますね．

```
10 FOR J=-0.5 TO -5 STEP -0.5
20 X=X+1
30 PRINT X,J
40 NEXT J

RUN
 1              -.5
 2              -1
 3              -1.5
 4              -2
 5              -2.5
 6              -3
 7              -3.5
 8              -4
 9              -4.5
 10             -5
```

式3を使った場合の例だ.

Xの値　　Jの値

行番号30で，XとJとの間がコンマになっているため，文字間隔が離れている．セミコロンにすると接近する．

なお，NEXT のあとの変数名は，省略することができます．また式1～式3は，ここでは数値で示しましたが，この段階で適正な数値になりえさえすれば，どんなに複雑な式であってもかまいません．事実，長いプログラムの中にこのステートメントを使うとき，以前の処理の結果次第で，ループのやり方を変えてやるような場合便利です．

では，次のプログラムを実行してみましょう．

```
10 INPUT X, Y, Z
20 FOR I=X TO Y STEP Z
30 PRINT"A";
40 NEXT I
```

こんな風に全部変数にしてやってもいいんだよ．

```
10 CLS:PRINT" ｳﾘｱｹﾞ ﾉ ｸﾞﾗﾌ"
20 FOR I=1 TO 40:PRINT"-";:NEXT
30 LOCATE 0,15
40 FOR I=1 TO 40:PRINT"-";:NEXT
50 LOCATE10,1:PRINT"┬"
60 FOR I=1 TO 13:J=J+1:LOCATE 10,J+1:PRINT"|":NEXT
70 LOCATE 10,15:PRINT"┴"
80 LOCATE 0,2:PRINT"ﾏｸﾞﾛ":PRINT"ﾊﾏﾁ":PRINT"ﾋﾗﾒ
100 LOCATE 11,2:FOR A=1 TO 12:PRINT"*";:NEXT
110 LOCATE 11,3:FOR A=1 TO 15:PRINT"*";:NEXT
120 LOCATE 11,4:FOR A=1 TO 8:PRINT"*";:NEXT
130 LOCATE 0,16
```

NEXTの あとの変数名は省略してある．

```
ｳﾘｱｹﾞ ﾉ ｸﾞﾗﾌ
----------------------------------------
ﾏｸﾞﾛ       |************
ﾊﾏﾁ        |***************
ﾋﾗﾒ        |********
           |
           |
           |
           |
----------------------------------------
```

行番号20から70までが，表のけい線を作るプログラムです．FOR～NEXT が，たくさん使われています．また，LOCATE で画面上の位置を指定しています．これらのコマンドを使って横線や縦線を描き出しているのです．

左は，このプログラムを実行した結果です．

ウリアゲ　ノ　グラフ　ができ上がりました．100～120 で，Aの数値を 12，15，8 としていますが，この値をたとえばINPUT 文によってキーインしてやり，キーイン次第で変えてやれば，よりよいプログラムが作れるでしょう．行番号130は，グラフの中にカーソルが入ると，じゃまになるので入れた命令です．

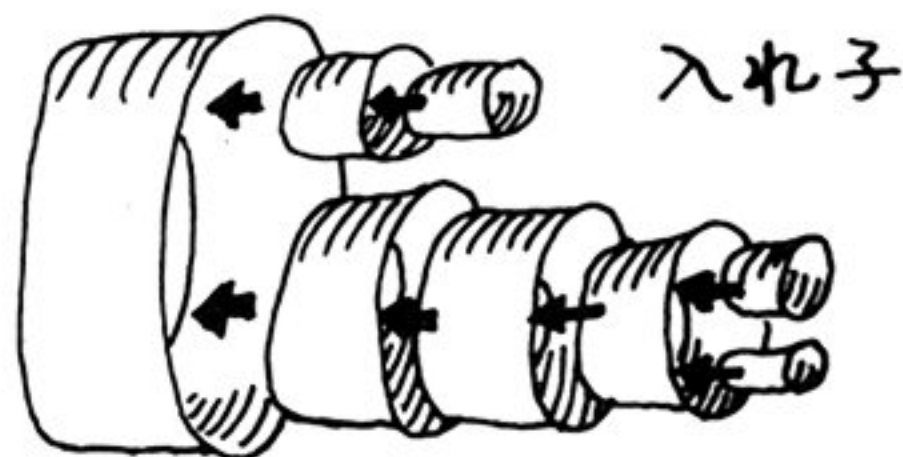
100～120 のように，似たプログラムが続くときは，行番号の100 を110 と修正したあとで，必要な場所だけ同様に修正し，RETURN キーを押すと，簡単にふやしていくことができます．

なお，ここではFOR～NEXT 文を使って，線を引いてみましたが，実際にはもっと便利な方法もあります．それはあとのお楽しみです．

2.10 多重ループ

FOR ～ NEXT のループがプログラムを作る上で，たいへん便利な命令だということは，もうおわかりいただけたことと思います．例の中に出てくる使い方は，簡単なものだけを示していますが，実際的なプログラムでは，もっと複雑で長いものに仕上げる場合も少なくありません．

さらに，何重にもループで囲んでやることもあります．なお，このように何重にもなる形をとるものを，入れ子の状態，ネスト（nest）ともいいます．

これは三重のループに囲まれた，FOR～NEXT ループの例です．＊1～＊5の部分に必要なプログラムを追加することができます．ここで気をつけて欲しいのは，一番外側のループはIという変数，真ん中がJ，内側がKという風に，お互いに交差しないように作られていなくてはならないということと，三つの変数はそれぞれ違った名前でなくてはならない，という点です．たとえば，次のようなループの作り方はルール違反であり，コンピュータは動いてくれません．

お互いに交わり合う作り方は，許されない

もう一つ気をつけていただきたいことは，たとえばGOTO文などで，FOR～NEXTのループの中から飛び出すことは可能ですが，よそにあるGOTO文などから，このループに飛び込むことは許されないということです．

ではここで，九九の計算をするプログラムを作ってみましょう．

```
10 CLS
20 FOR X=1 TO 9
30 FOR Y=1 TO 9
40 Z=X*Y
50 LOCATE(X-1)*3,Y:PRINT Z
60 NEXT Y,X
```

行番号60でY，Xの順序に気をつけてください．行番号50は，九九の計算結果をきちんと並べるための手立てと，結果を表示するための命令です．なぜ，こうするときちんと並ぶかは考えてください．LOCATE A，Bは，Aが横，Bが縦の座標でしたね．

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

◆ これは，多重のループではありませんが，やはりFOR～NEXTを使った楽しい問題です．

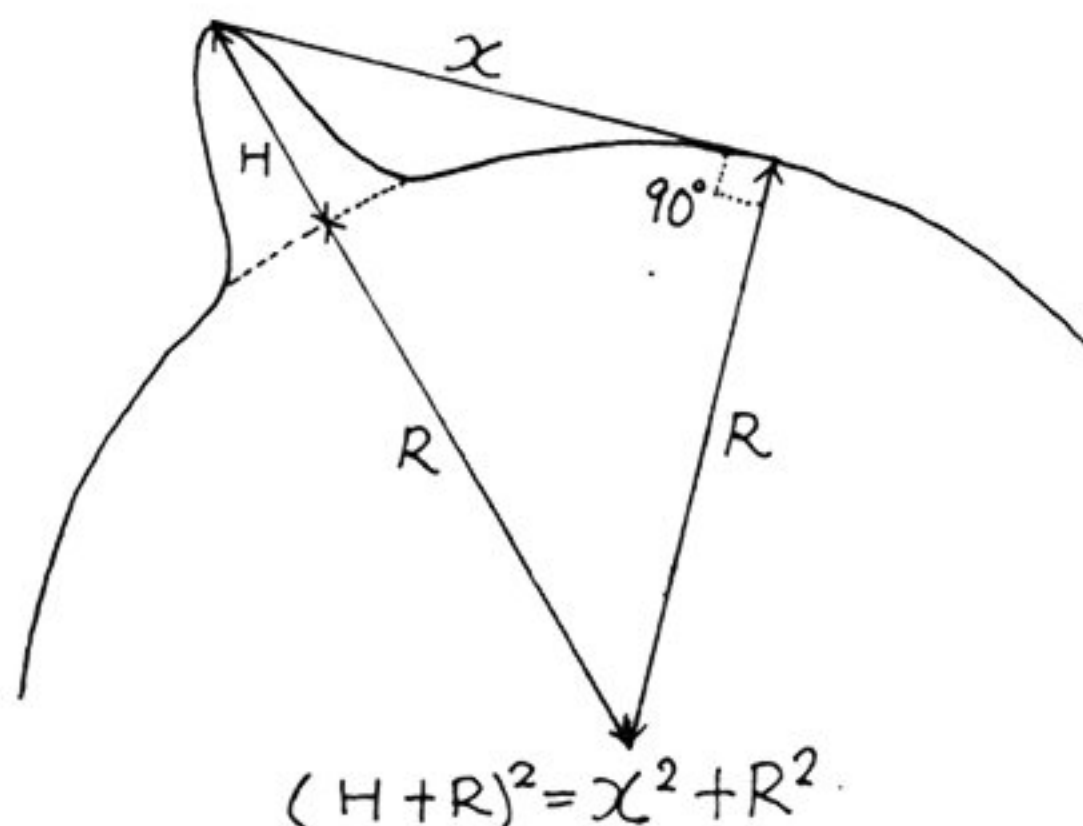
もし途中に，山や建物などの視界をさえぎるものがなかったとしたら，日本一の山，富士山は直線距離にして220～230 km離れた金沢からは見えるでしょうか？　約100km離れた東京からは，何合目以上が見えるでしょうか？　名古屋なら？　もし，富士山の位置にエベレスト山があったら？　さらに地球上ではなく木星だったら？

山の高さや，離れている距離，それに天体の半径などは，地図や文献を調べて，人の手によって入れてやるにしても，コンピュータなら簡単にやってのけます．

それ以前に，例によってコンピュータにどんな仕事をやらせようとするのかを，考えなくてはなりません．図を参照してください．図からわかるように，**三平方の定理**を使って，

$$X^2+R^2=(H+R)^2$$

$$X^2+R^2=H^2+2HR+R^2$$

$$X^2=H^2+2HR \quad \text{H}^2\text{は無視できるほど小さい}$$

天体	赤道半径(km)	山	高さ(m)
地球	6378	富士山	3776
月	1738	駒ヶ岳	2132
太陽	696000	白根山	3192
水星	2439	穂高岳	3190
金星	6052	槍ヶ岳	3180
火星	3397	エベレスト	8848
木星	71400	モンブラン	4807
土星	60000	キリマンジャロ	5895

したがって $X^2=2HR$

$$X=\sqrt{2HR}$$

この計算を，コンピュータにやらせればよいわけです．ここで $\sqrt{\quad}$（平方根）という，コンピュータにはない記号が出てきましたね．安心してください．これにかわるものがちゃんとあるのです（ほかにどんな類似のものがあるかは後述します）．SQR（式）がそれです．

では，プログラムを作ってみましょう．この程度のプログラムなら，もし詰め込んだら1行か2行で書くことも可能でしょう．1行に書いても，FOR～NEXTはちゃんと実行してくれます．ここではわかりやすくするため，何行にも分けて示します．

行番号60のR/1000はmをkmの単位に直す計算です．何かのキーを押すと，また行番号10にかえります．

プログラム

```
10 CLS
20 INPUT"テンタイ ノ ハンケイ ハ (km) ",R
30 INPUT"ヤマ ノ タカサ ハ (m) ",H
40 FOR N=1 TO 10
50 PRINT N;"ゴウメ"
60 X=SQR(2*(R/1000)*H*N/10)
70 LOCATE 12,1+N
80 PRINT X;"km"
90 NEXT N
100 INPUT I$
110 GOTO 10
```

RUN

```
テンタイ ノ ハンケイ ハ (km) 6378
ヤマ ノ タカサ ハ (m) 3776
 1 ゴウメ      69.4022 km
 2 ゴウメ      98.1495 km
 3 ゴウメ      120.208 km
 4 ゴウメ      138.804 km
 5 ゴウメ      155.188 km
 6 ゴウメ      170 km
 7 ゴウメ      183.621 km
 8 ゴウメ      196.299 km
 9 ゴウメ      208.207 km
10 ゴウメ      219.469 km
?
```

地球の半径

富士山の高さ

何かの文字を入れるとつぎの行番号110を実行する．

2.11 IF～THEN～ELSE

コンピュータが比較・判断し，その結果次第で異なる仕事をするという，いかにもコンピュータらしい命令の一つです．こう書くとむずかしそうですが，何でもない命令です．もし～ならば～です．という簡単な判断なのですから．

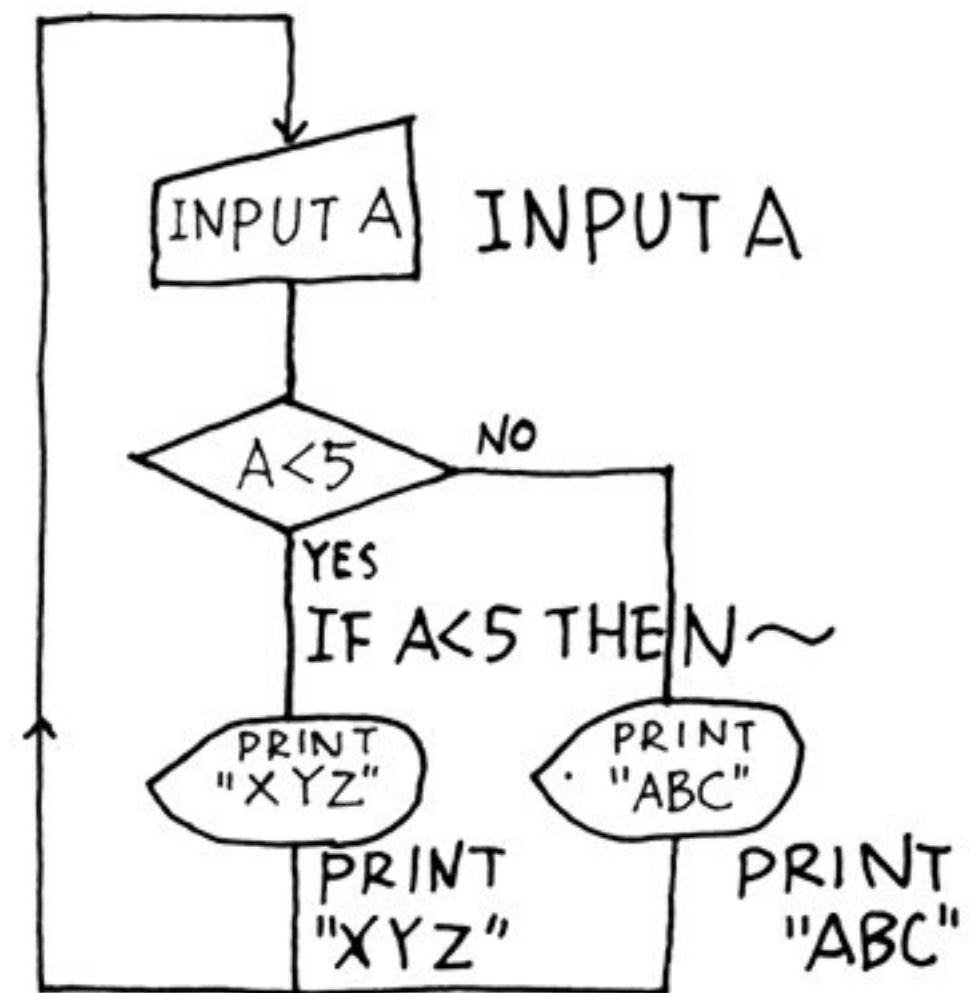
IF 式 THEN {文または行番号}
[ELSE {文または行番号}]

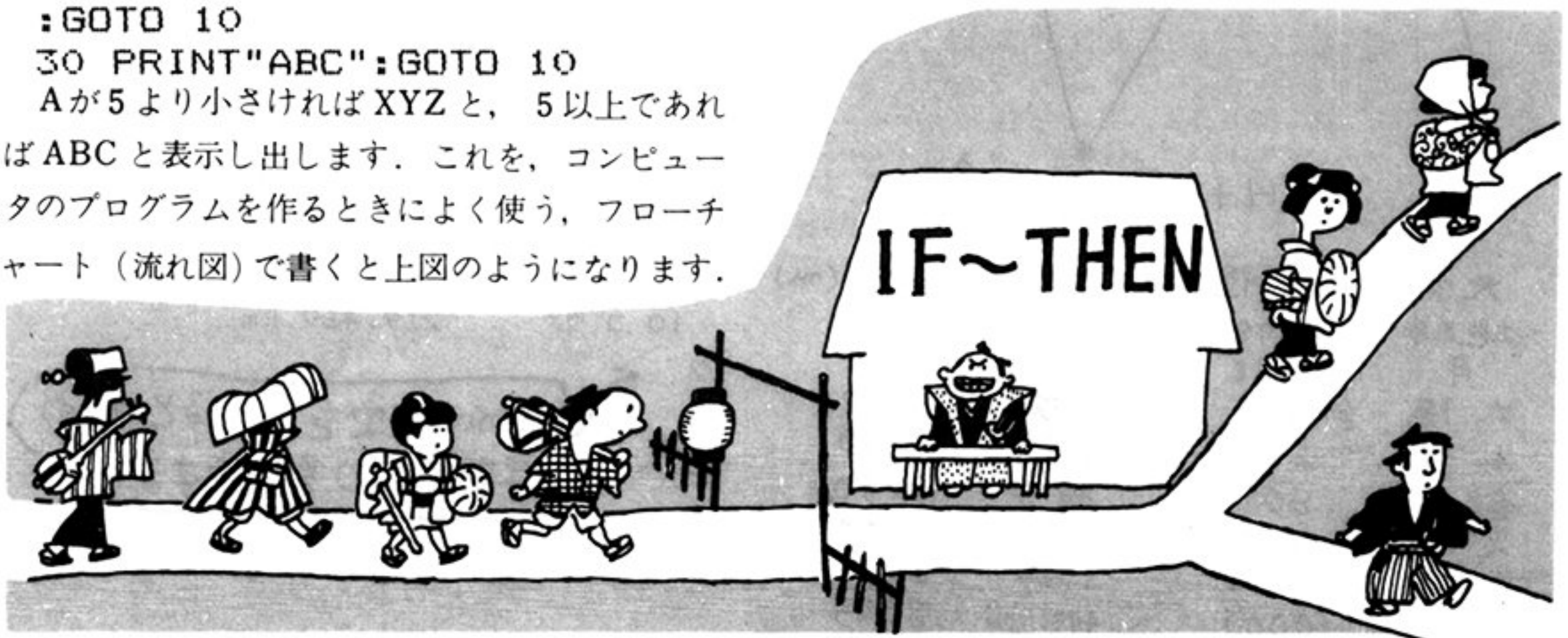
とにかく例を調べてみましょう．

```
10 INPUT"ｷﾐ ﾉ ﾄｼ ﾊ ?",T
20 IF T<20 THEN PRINT"ﾐｾｲﾈﾝ"
30 IF T=20 THEN PRINT"ｾｲｼﾞﾝ"
40 IF T>20 THEN PRINT"ｵﾄﾅ"
```

何でもないプログラムですから，もうほとんど説明の必要はないと思います．T＞20，T＝20，T＜20が正しければ（TRUE）THENのあとに続く文を実行しなさい，という意味です．もし，ELSE（でなければ）がついていたら，このあとに続く文を実行します．もし文のところが行番号になっていたら，その行に飛んで行きます．また，ELSEがなくて，IF文THEN～だけのとき，文の結果が真でなければ，THENのあとに続く文を無視して，次の行に進みます．

```
10 INPUT A
20 IF A<5 THEN PRINT"XYZ"
:GOTO 10
30 PRINT"ABC":GOTO 10
```

Aが5より小さければXYZと，5以上であればABCと表示し出します．これを，コンピュータのプログラムを作るときによく使う，フローチャート（流れ図）で書くと上図のようになります．

```
10 A=A+1
20 PRINT"ABC"
30 IF A<10 THEN 10
40 END
```

ABCというのを10回表示してくるはずです．ちょうどFOR～NEXTと同じ働きをさせることもできるわけです．ENDは，プログラムの実行を中止させるための命令で，プログラムのどこにあってもかまいません．上のように一番終わりにある場合は，省略してもいっこうにかまいません．また，THENのあとが行番号であれば，THENのかわりにGOTOであってもかまいません．

```
10 INPUT A
20 IF A<7 THEN
      IF A>4 THEN PRINT "XYZ"
      ELSE PRINT "UVW"
   ELSE PRINT "ABC"
30 GOTO 10
```

行番号20で，THENのあとに，またIF文が出てくる例を示してみました．この部分は特に4行に書いてありますが，これはその構造をわかりやすくするためで，実際には1行に書いてもよいのです．多重のFOR～NEXTの場合のように，ネスト（入れ子）の状態になっていますね．とにかくRUNさせてみましょう．

```
RUN
? 8 ……… A<7でないのでABCが出る.
ABC
? 6 ……… A>4なのでXYZが出る.
XYZ
? 2 ……… A>4でないのでUVWが出る.
UVW
```

◆ ここで出てくる式というのは，A＝B＋Cのような算術式とは，ちょっと違います．このように<，>，＝のような記号（関係演算子という）で判断し，その判断結果がYESかNOかで出てくるものを関係式といいます．IF文でいう式には，関係式のほかに，論理式にも使えます．論理式については後述します．関係演算子には，次のようなものがあります．

＝	等しい
<>，><	等しくない
<	小さい
>	大きい
<=，=<	等しいか小さい
>=，=>	等しいか大きい

◆ INKEY $

ここで，INPUT文とちょっと似ているINKEY $という，便利な命令をおぼえておきましょう．

```
10 A$=INKEY$
20 IF A$="" THEN 10
30 PRINT A$
40 GOTO 10
```

キーボードから好きな文字をキーインしてやるごとに，CRT上にその文字が表示されます．別にどうというプログラムではないのですが，この中ではIF～THENのステートメントがちゃんと働いているのです．

INKEY $は，キーボードから文字を読み取る命令であり，行番号20でその文字が″″，つまり空文字であれば行番号10に戻るようになっています．簡単な命令ですが，実用上はいろいろと便利な使い方ができます．

INKEY$をIF文を使わないで利用するとどうなるでしょう

```
10 A$=INKEY$
20 PRINT A$
30 GOTO 10
```

または

```
10 A$=INKEY$
20 PRINT A$;
30 GOTO 10
```

実際に試してみてください．

IF A+B=C+D THEN PRINT "X"

関係式

これは算術式ではない．比較してこの式が成立するかどうかを調べてるんだ

2.12 WHILE～WEND

FOR～NEXT の場合は，ループの中を少なくとも1回は通ります．

```
10 FOR I=1 TO 1
20 PRINT "XYZ"
30 NEXT I
RUN
XYZ
```

WHILE 文は，FOR～NEXT によく似たループでありますが，IF～THEN の場合のように，論理式または関係式によって判断し，その結果が真である間だけループを回ります．そのため初めから真ではない場合には，このループを1回も通らず，素通りすることもあるのです．

WHILE 式 ～ WEND

```
10 INPUT Z
20 WHILE Z>=10
30 Z=Z-1
40 PRINT"XYZ";Z
50 WEND
60 GOTO 10
```

これを実行すると，まずZの値をたずねてきます．そこで1桁の数字を入れると，再度？が出ます．しかし，2桁の数字を入れると，その値から10を引いた数だけXYZが表示されるはずです．1桁の数字だと，行番号20が真でなくなり，素通りするからですね．2桁だと，この結果が真となり，30～40の仕事を実行するわけです．そして，この間にZから1を引く仕事と，XYZという文字列と，Zの値を表示する仕事をします．

2.13 REM

長いプログラムを作る場合，いろいろな仕事をする部分が出てきます．そこで，プログラムの実行には関係ないが，何か注釈文を入れておきたい，というような場合が少なくありません．そんなとき利用するのがREM文です．REMのあとに続く文字は，プログラムの実行には関係しません．

```
10 REM:オワリダ゛ゾ゛!STOP!
20 PRINT"XYZ":'ヤメロ!
```

たとえENDと書こうが，：でつないでオワリダゾ！と強調しようが，コンピュータは知らん顔で仕事を続けます．なお，REMのかわりにアポストロフィ’で代用することもできます．

◆ 関係演算子を使った式で，ここでは単に数値だけの例をあげましたが，両辺には算術演算子を使った式があってもかまわないわけです．そこで問題になるのが，演算子の優先順位です．もちろん，(　)は全てに優先します．

1　∧
2　－（マイナス符号）
3　＊，／
4　MOD
5　＋，－
6　関係演算子（＝，＜＞，＞＜，＞，＜，＜＝，＝＜，＞＝，＝＞）
7　このあとに論理演算子が続きます．

A=1 : B=2 : C=3 : D=4 : E=5
なら
A+B*(C+D)+E∧B と A*B*C+D*E
とはどちらが大きい？

◆ プログラムの途中で BREAK キーを押したり，STOPまたはENDの命令にぶつかると，プログラムはそこで一時中止してしまいます．このようにして，プログラムを一時中断させてから，変数の中味や途中の経過，プログラムの内容などを調べたりすることができます．また，停止させたプログラムを再開させたいときは，CONTまたはGOTO～と入れてください．その後を実行させることができます．ただし，中断中のプログラムを変更したときは再開できません．なお，～にはSTOP文の次の行番号が，また BREAK キーで停止したときは，停止した行番号が入ります．

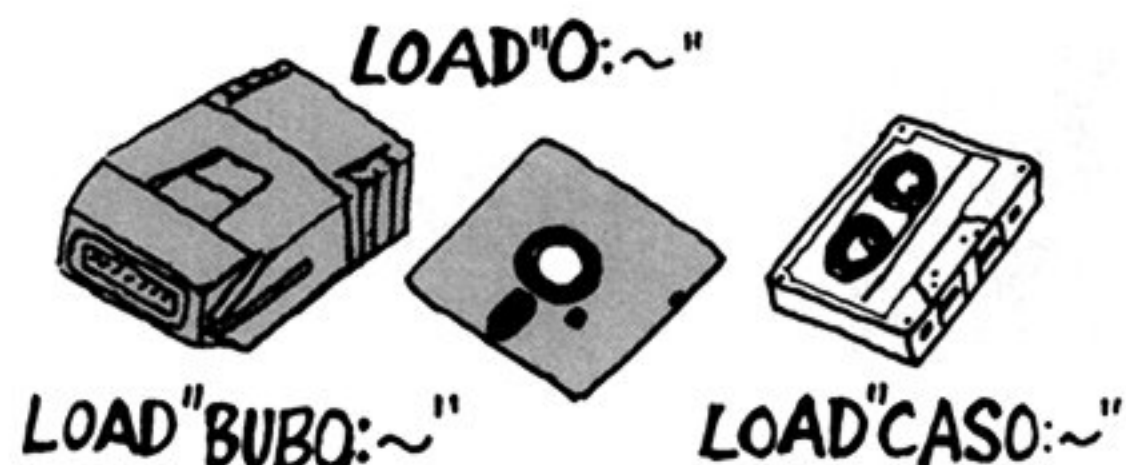
途中でプログラムを停止させ，変数の状態などを調べるのに便利．

2.14 入出力装置とのやり取り

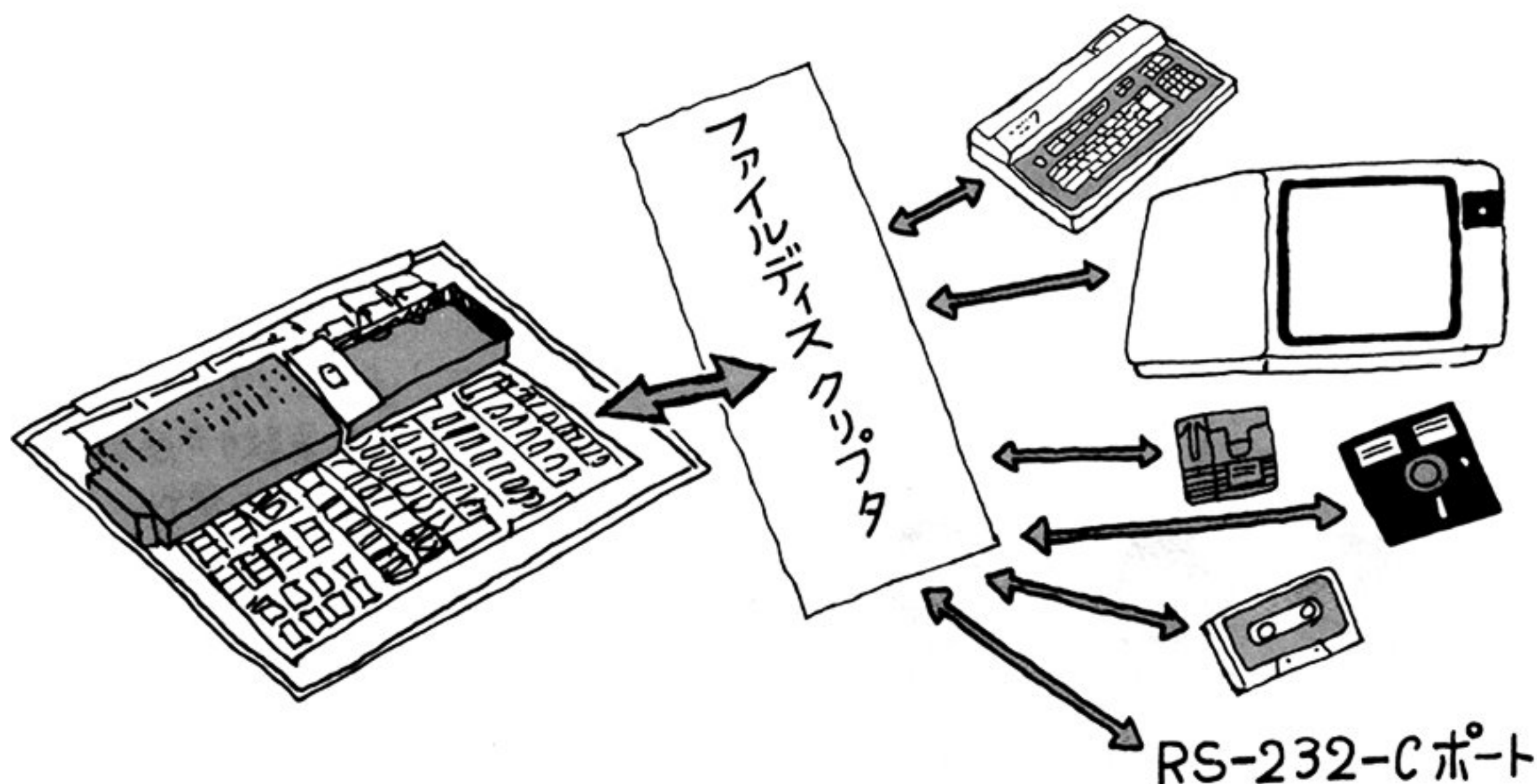
コンピュータは，せっかく作ったプログラムでも，いったん電源を切ってしまうと，完全に忘れてしまい，それでおしまいです．そこで，作ったプログラムを外部記憶装置に記憶させておき，必要に応じて再びコンピュータに伝えてやる方法がとられます．長いプログラムを作る場合，途中の要所要所で，万一の場合全てが消えてしまわないように，記録を取っておくのも賢明なやり方でしょう．

こんなとき，まずおぼえておいていただきたいのが，ファイルディスクリプタ（file descripter: ファイルの区別）という言葉です．F－BASICでは，全ての入出力装置をファイル（書類差し）という概念で扱っています．そこで，入出力装置との情報のやり取りには，ファイルディスクリプタを入れて，どの装置（デバイス）との間なのかをはっきりさせてやる必要があります．

ファイルディスクリプタは，次の形式になっています．

〃[**デバイス名**][[(**オプション**)]
[**ファイル名**]]〃

デバイス名，オプション，ファイル名は，それぞれ省略してもよい場合があります．オプションは，6文字以内の英数字をかっこでくくって指定します．

ファイル名は，8字以内，文字列の中に：や〃があってはいけません．代表的な入出力装置のデバイス名は次のとおりです．

入出力装置名	デバイス名	入力	出力
キーボード	KYBD：	○	×
スクリーン	SCRN：	×	○
プリンタ	LPT0：	×	○
RS－232Cポート 0	COM0：	○	○
1	COM1：	○	○
2	COM2：	○	○
3	COM3：	○	○
カセットテープ	CAS0：	○	○
フロッピーディスク 0	0：	○	○
1	1：	○	○
2	2：	○	○
3	3：	○	○
バブルカセット 0	BUB0：	○	○
1	BUB1：	○	○

あなたの作ったプログラム（メインメモリに記憶されている）を，カセットテープに記録する操作をやってみましょう．使う命令は次の形式です．

SAVE "ファイルディスクリプタ"

まず，あなたのプログラムに，ファイル名を付けてください．それを"ABC"としましょう．カセットテープレコーダはCAS0：です．最後の0はオーではなく，ゼロだということに注意してください．テープレコーダをセットして，録音状態にキーをセットし，上記表からわかるように，

SAVE "CAS0：ABC" RETURN とすると，テープレコーダは記録し始めます．完了は，Readyが出るのですぐわかります．これであなたのプログラムは，安全になったわけです．でもちょっと心配ですね．苦労して作ったのが，万一正確に記録されたかどうか——．

こんなとき，正確かどうかを確かめる命令が，LOAD？です．

LOAD？［"［CAS0：］ファイル名"］

ファイルディスクリプタの指定を省略すると，テープの先頭のファイルと比較照合されます．

LOAD？"CAS0：ABC" RETURN

テープを先頭のところまで巻き戻し，カセットテープレコーダの再生キーを押してください．照合が始まります．そして正しく記録されていないときは"Device I/O Error"と出ます．

OKでしたらこれで電源を切ろうと，今までのプログラムを全部消してしまっても，安全です．念のためにもう一度，この記録されたプログラムをコンピュータに入れてみましょう．

RUN "ファイルディスクリプタ"

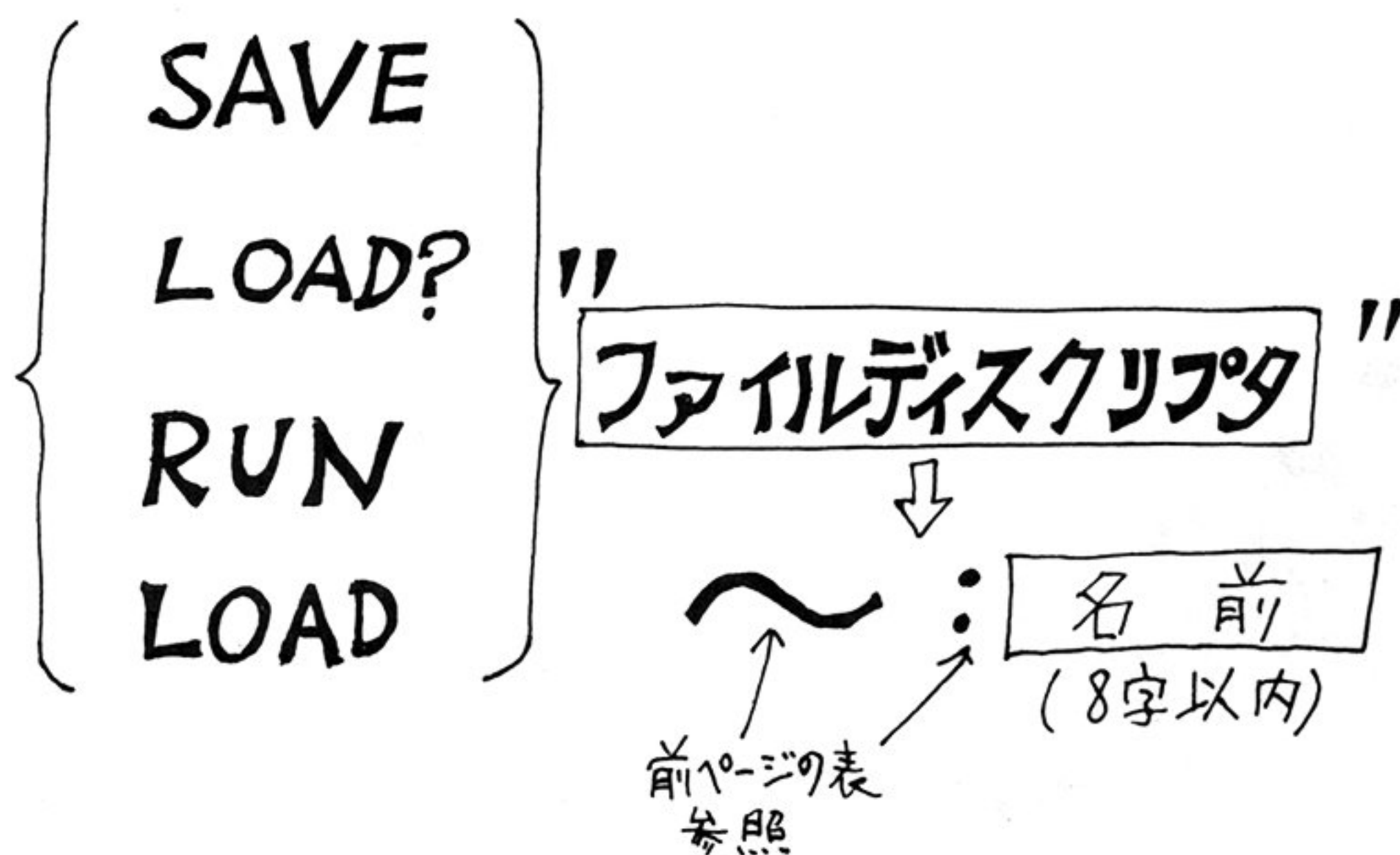
メモリにインプットした後，ただちに実行します．

LOAD "ファイルディスクリプタ"

メモリにインプットします．このままでは実行しません．

カセットテープを巻き戻し，プログラムを入れ直してみてください．

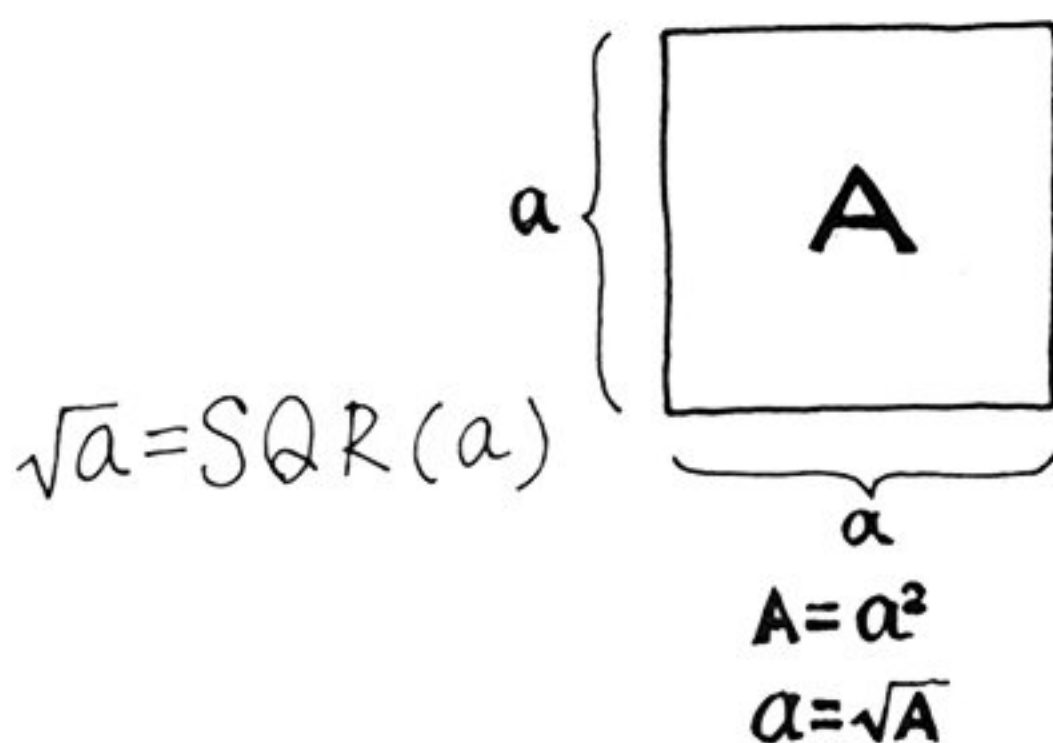
RUN "CAS0：ABC" RETURN

で，プログラムがロードされた後，実行を開始するはずです．

2.15 数値関数

たとえば，平方根の計算をする命令は，SQR（X）でした．Y＝SQR（X）はYを2回掛けるとXの値になる数，という意味ですね．もし，この計算をSQRという簡単な命令でできないと，かなり面倒な演算式をいちいち作ってやらなくてはなりません．F－BASICでは，このようによく利用する関数（数値関数）は，簡単な命令によってすぐ実行できるようになっています．

◆ **SQR（式）**：Square（平方根）の略です．

SQR（2）＝1.41421　　SQR（10）＝3.16228

演算は単精度で実行されます．

◆ **SGN（式）**：式の値が正なら1，0なら0，負なら－1になる関数です．

SGN（5）＝1　　SGN（0）＝0

SGN（－5）＝－1

$$\log_{10} x = \frac{\log_e x}{\log_e 10}$$

$$\log_e 10 = 2.30259$$

◆ **LOG（式）**：式の自然対数の値を出します．なお，式の値は正数でなくてはならず，また演算は単精度で実行されます．

LOG（2）＝.693147　LOG（10）＝2.30259

$$X = EXP(Y)$$

$$X = e^Y$$

$$e = 2.71828$$

$$e^2 = 7.38906$$

◆ **EXP（式）**：*e*（*e*＝2.71828）を底とした指数関数の値を与えます．式の値は，87.3366未満でなくてはなりません．演算は単精度で実行されます．

EXP（2）＝7.38906　　EXP（10）＝22026.5

◆ **RND（式）**：0と1との間の乱数を得ることができます．式の値が正であれば，同じ乱数の系列の負の乱数を与え，0の場合は一つ前に発生した乱数を，そして負なら新しい乱数の系列を作り出します．乱数はコンピュータ処理にいろいろと利用される数です．あとでもう一度調べ直すことにしましょう．

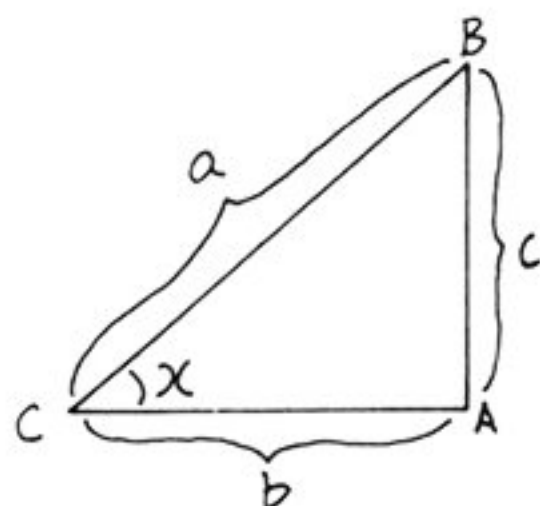
```
10 FOR I=1 TO 5
20 PRINT RND(I);RND(0);RND(-I)
30 NEXT

Ready
RUN
 .591065   .591065   .522223
 .958935   .958935   .522223
 .958935   .958935   .772223
 .208935   .208935   .522223
 .958935   .958935   .147223
```

◆ **ABS(式)**：absolute（絶対値）の略です．

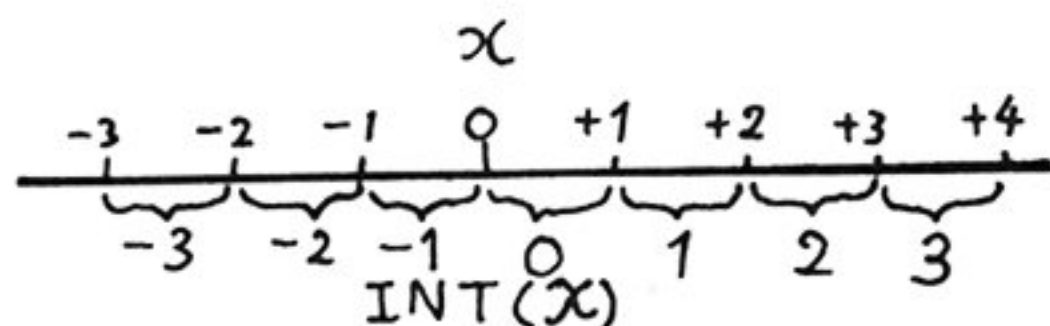
数学でいうと $|ax^2+b|$ と同じ意味，つまり，マイナスの数もプラスにしてしまいます．ABS(−3.5)は3.5ですね．

◆ **SIN(式)**：数学の $\sin x$ の計算をします．ただし，式の単位はラジアンです．°(度)ではない点に気をつけてください．もし°(度)の式のSINを計算したかったら，$\pi/180$ を掛けて，ラジアンに直してやる必要があります．ここで π は3.14159ですから，0.0174533 を掛けてもかまいません．

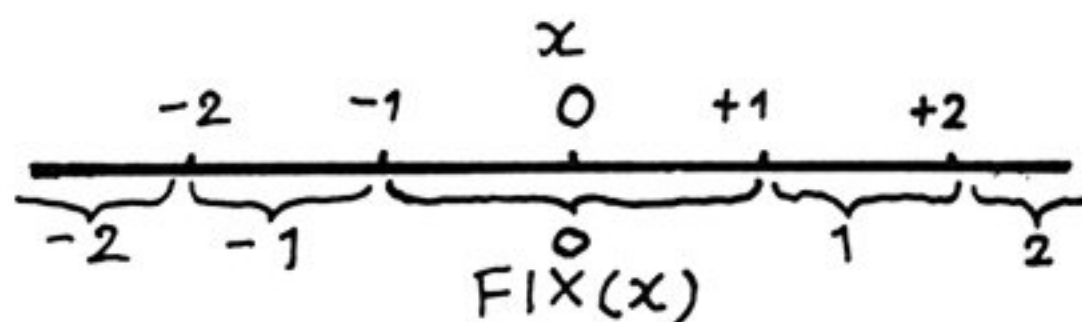
なお，SINなどの三角関数の演算は単精度で実行されます．三角関数では，このほかにCOS(式) TAN(式)，ATN(式)などがあります．ATN(式)は $\tan^{-1}$ 式のことです．

度	ラジアン
0°	0
180	π
x	3.14159＊x/180 または0.0174533x

◆ **INT(式)**：式の値を，この数を超えない最大の整数に変えてしまいます．

INT(6.4)＝6　　INT(−6.4)＝−7

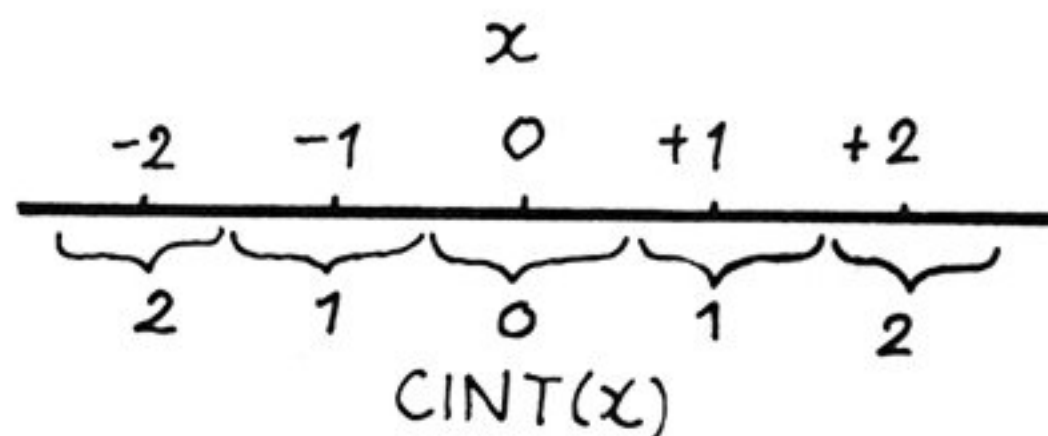
INTはinteger（整数）の略です．

◆ **FIX(式)**：式の値の整数部分を取り出します．

FIX(6.4)＝6　　FIX(−6.4)＝−6

FIXは，固定するとか，整えるという意味です．

◆ **CINT(式)**：式の値の小数部分を四捨五入して，整数に変えてしまいます．

CINT(6.4)＝6　　CINT(−6.4)＝−6

CINT(6.5)＝7　　CINT(−6.5)＝−7

ただし，式の値が −32768 から 32767 までの範囲でないとエラーになります．

◆ **CSNG(式)，CDBL(式)**：CSNGは，式の値を単精度形式の数値に，またCDBLは式の値を倍精度形式の数値に変えてしまいます．

CSNG(1234567890)＝1.23456E＋09

CDBL(1234567890)＝1234567890

SingleかDoubleかの違いというわけです．

2.16 三角関数

cos：コサイン

tan：タンジェント

三角関数は，コンピュータ処理によく使われる関数の一つです．そこで，三角関数にあまり親しみのない方のために，簡単にふれておきます．よくご存知の方は，飛ばしてくださって結構です．

直角三角形 ABC があるとき，$\sin\theta$，$\cos\theta$，$\tan\theta$ は次のような関係にあります．

$$\sin\theta=\frac{BC}{AB}$$

$$\cos\theta=\frac{AC}{AB}$$

$$\tan\theta=\frac{\sin\theta}{\cos\theta}=\frac{\frac{BC}{AB}}{\frac{AC}{AB}}=\frac{BC}{AC}$$

これは，X－Y座標軸の交点を中心とし，半径1の円を一定の角速度で描くと，図のような波形のカーブになります．このカーブがサインカーブです．横軸は，時間の経過と考えてください．

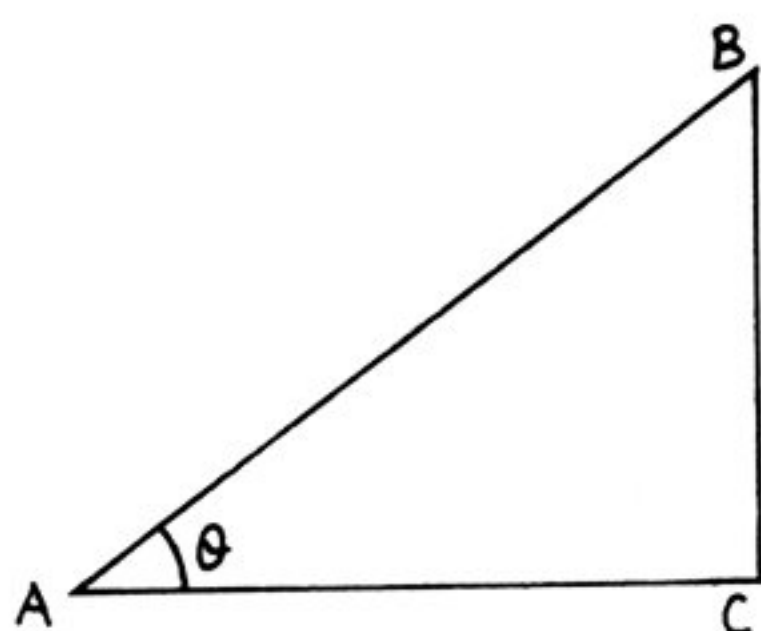
$\sin\theta_x=x$，$\cos\theta_y=y$，$\tan\theta_z=z$ とするとき，逆の関係になっている関数をアークサイン，アークコサイン，アークタンジェントといい，次のように表わします．

$$\sin^{-1}x=\theta_x,\quad \cos^{-1}y=\theta_y,\quad \tan^{-1}z=\theta_z$$

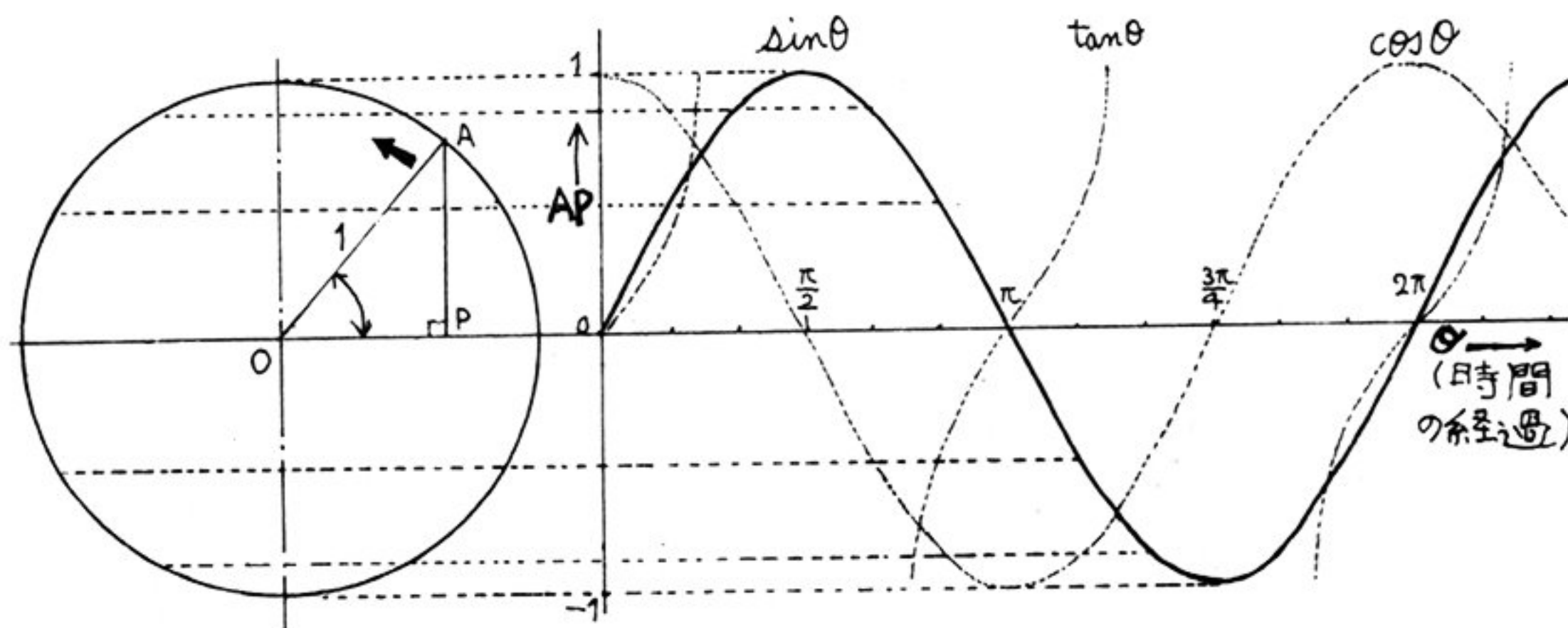
なお，BASIC で使う場合には，θ はラジアンでなくてはなりません．

では，この関数を使って，正弦波（サインカーブ）を描く方法を考えてみましょう．

右側の図がサインカーブです．この曲線は，左側のような，半径が1の円の中にある直角三角形 OPA で，OA が矢印の方向に回転して行く場合，A P の長さを縦軸に，横軸に θ を取って描いた線です．

同様にして $\cos\theta$，$\tan\theta$ を取ると，図のようになります．

$$\sin\theta=\frac{BC}{AB}$$

$$\cos\theta=\frac{AC}{AB}$$

$$\tan\theta=\frac{BC}{AC}$$

2.17 スクリーン・エディタ(その2)

1.10で示したキー操作の続きです．プログラム作成を効率よく進めるために，これらの機能を体得してください．

◆ DEL キーによく似た働きをするのが ⇦ キーです． ⇦ キーはカーソルが一番左寄りにあるときは，何の働きもしません．表示されている文字の上を， → キーを使って少しばかり右の方に移動してから，押してみてください．カーソルと重なった文字から，右側の文字が左側に移動し，今まであったカーソルの左側の文字がつぎつぎに消えてしまいます．長く押し続けると，どんどん左側の文字を消してしまい，その分だけ右側の文字が，左側に寄ってきます．なおFM－8では BS キーになっています．

このようなキーは，プログラムを修正したり削除したりするとき，非常に有効にあなたの手助けをするに違いありません．また， CTRL キーを押しながら H キーを押しても，全く同様な働きをします．

⇦ ＝ CTRL ＋ H

◆ カーソルのキーのある文字も含めて，それより右側以後の文字を全部消してしまうのが EL キーです．不要な命令を消してしまうとき，たいへん便利です．

EL ＝ CTRL ＋ E

◆ 一定の文字数ずつ右側に飛びながら，飛んだ部分の文字を消してしまうのが TAB キーです．BASICでは，8文字ごとに飛びます．

TAB ＝ CTRL ＋ I

◆ **EDIT**

これは，キー操作ではありませんが，やはりプログラム作成のとき便利なコマンドです．

EDIT　行番号

EDIT文を実行すると，画面がクリアされてきれいになり，指定された行のプログラムが表示されて，カーソルが行番号の直後で点滅します．

◆ **HARDC**

プリンタが接続されているとき，CRT上に表示されているデータを，プリンタで打ち出してくれます．

HARDC ［0または1または2］

もし，あとに続く数字が0または何もないときは，キャラクタ（キーから入れることのできる文字類）だけを表示します．また1のときは，ドット（点）の表示もしますが，横方向の1ドット分を4ドットに拡大して，プリントします．
ただし，

青，赤，緑，白のドット……黒でプリント
紫，水色，黄色は……………灰　〃
黒のドットは……………………プリントされない

また2のときは，画面上の1ドットをやはり1ドットでプリントします．ただし，色の付いたドットは，このように区別せず，全て黒で出ます．

3 画面制御・グラフィック機能

CRTに写し出したり，プリンタで紙に印刷し出すときの文字や図形の，いろいろな処理のし方について述べる章です．

他の章の場合もそうですが，例にあげているプログラムは，いずれも内容を理解していただくためのものです．特に楽しいものではありませんが，これらの命令を使った応用プログラムは，あなたの腕次第，美しい図形，楽しい図形を作ってください．

3.1 WIDTH

CRT画面上に書ける文字の総数を決める命令で，その形式は次のとおりです．

```
WIDTH ［1行の文字数］
      ［,［1画面の行数］］
```

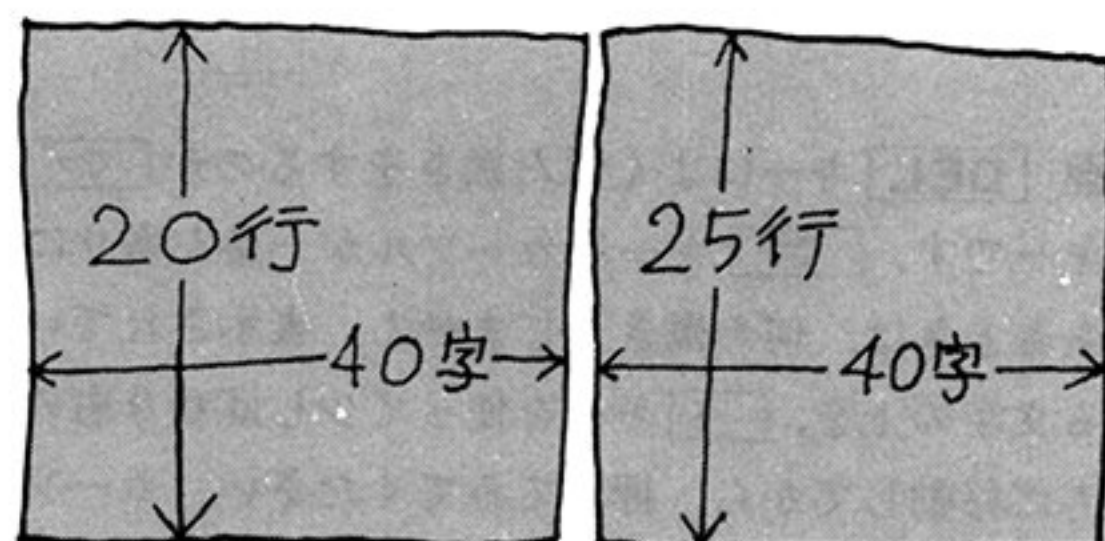
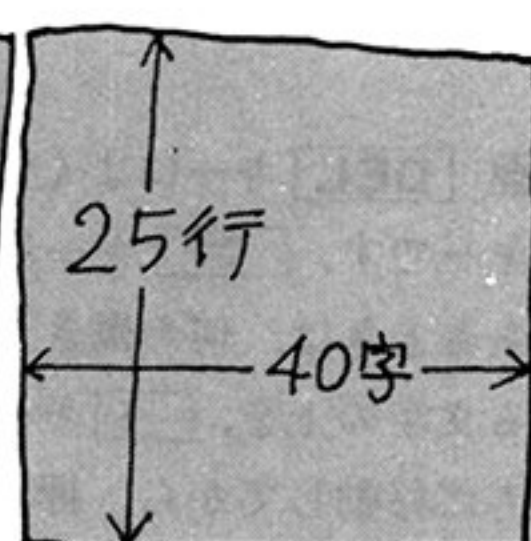
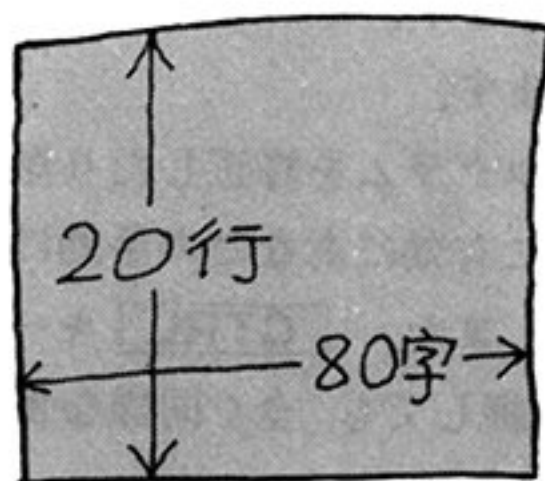
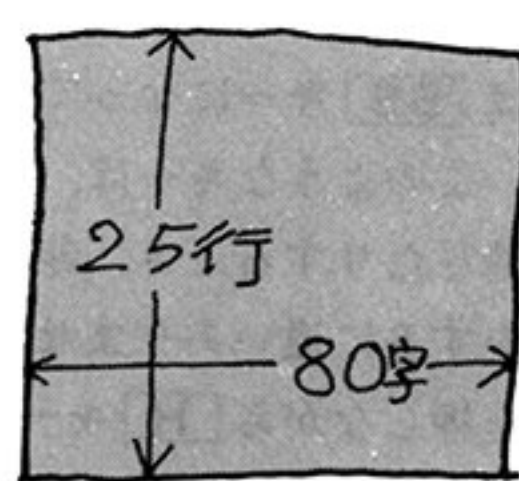
BASICが起動したとき，CRT上には40字×20行（FM−11は80字×25行）の文字が出るようになっています．しかし，文字の大きさが変わってもいいから，もっと違った大きさあるいは行間隔で表示させたい，というときに利用するのがこの命令です．BASICでは，次の種類を選ぶことができます．

WIDTH 40，20………40文字，20行
WIDTH 40，25………40文字，25行
WIDTH 80，20………80文字，20行
WIDTH 80，25………80文字，25行

図1　図2

WIDTH80,20　WIDTH80,25

図3　図4

もちろん，処理の結果の文字やグラフィック記号を変えてやるために，WIDTH命令をプログラムの中に使うことはできますが，ここではまず直接モードによって，実際に上の命令を入れてみてください．画面がクリヤーされ，Readyとあなたが指定したWIDTHにしたがって表示されます．次ページの図は，このようにして文字数や行数を設定した後，プログラムを表示させ，ハードコピー（CRTに出ている内容を印刷させる．HARDC2命令）をとったものと，そのときのディスプレイ画画の写真を示します．

なお，この命令を実行すると，次の章で出てくるCONSOLE命令は，コンソールカラースイッチの部分を除いて解除されます．たとえばWIDTH，20を実行すると，同時にCONSOLE 0，20，0が実行されます．また，FM−11の場合のWIDTH命令には，上記以外にもいろいろな機能がありますが，その詳細については，F−BASIC 文法書を参照してください．

WIDTH40,20

```
10 CLS
20 WIDTH 40,20
30 LOCATE 15,2
40 PRINT"ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ"
50 PRINT
60 FOR I=1 TO 7
70 COLOR I
80 FOR J=1 TO 22
90 PRINT J;
100 NEXT J
110 NEXT I
```

WIDTH40,25

```
10 CLS
20 WIDTH 40,25
30 LOCATE 15,2
40 PRINT"ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ"
50 PRINT
60 FOR I=1 TO 7
70 COLOR I
80 FOR J=1 TO 22
90 PRINT J;
100 NEXT J
110 NEXT I
```

WIDTH80,20

```
10 CLS
20 WIDTH 80,20
30 LOCATE 15,2
40 PRINT"ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ"
50 PRINT
60 FOR I=1 TO 7
70 COLOR I
80 FOR J=1 TO 22
90 PRINT J;
100 NEXT J
110 NEXT I
```

WIDTH80,25

```
10 CLS
20 WIDTH 80,25
30 LOCATE 15,2
40 PRINT"ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ"
50 PRINT
60 FOR I=1 TO 7
70 COLOR I
80 FOR J=1 TO 22
90 PRINT J;
100 NEXT J
110 NEXT I
```

```
LIST

10 CLS
20 WIDTH 40,20
30 LOCATE 15,2
40 PRINT"ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ"
50 PRINT
60 FOR I=1 TO 7
70 COLOR I
80 FOR J=1 TO 22
90 PRINT J;
100 NEXT J
110 NEXT I

Ready
```

WIDTH 40, 20の場合

```
LIST

10 CLS
20 WIDTH 40,25
30 LOCATE 15,2
40 PRINT"ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ"
50 PRINT
60 FOR I=1 TO 7
70 COLOR I
80 FOR J=1 TO 22
90 PRINT J;
100 NEXT J
110 NEXT I

Ready
```

WIDTH 40,25 の場合

```
LIST

10 CLS
20 WIDTH 80,20
30 LOCATE 15,2
40 PRINT"ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ"
50 PRINT
60 FOR I=1 TO 7
70 COLOR I
80 FOR J=1 TO 22
90 PRINT J;
100 NEXT J
110 NEXT I

Ready
```

WIDTH 80, 20の場合

```
LIST

10 CLS
20 WIDTH 80,25
30 LOCATE 15,2
40 PRINT"ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ"
50 PRINT
60 FOR I=1 TO 7
70 COLOR I
80 FOR J=1 TO 22
90 PRINT J;
100 NEXT J
110 NEXT I

Ready
```

WIDTH 80, 25 の場合

3.2 CONSOLE

キーボードから入力した命令文，あるいはデータは RETURN ごとに，あるいは文字数が1行分いっぱいになるごとに，1行ずつ追加されますが，CRT上にはWIDTHで指定された行数までしか，一画面同時には表示できません．それ以上の行数を入れると，画面の下から新しいものが入り，その分だけ一番上の行が画面から消えていきます．このことをスクロールといいます．Scrollは巻物のことで，巻物を巻いていくような感じになるからです．

CONSOLEは，スクロールする範囲，すなわちスクロールウインドウの大きさを指定する命令で，その形式は次のとおりです．

CONSOLE ［スクロール開始行］
［,［スクロール行数］
［,［ファンクションキー表示スイッチ］
［,コンソールカラースイッチ］］］

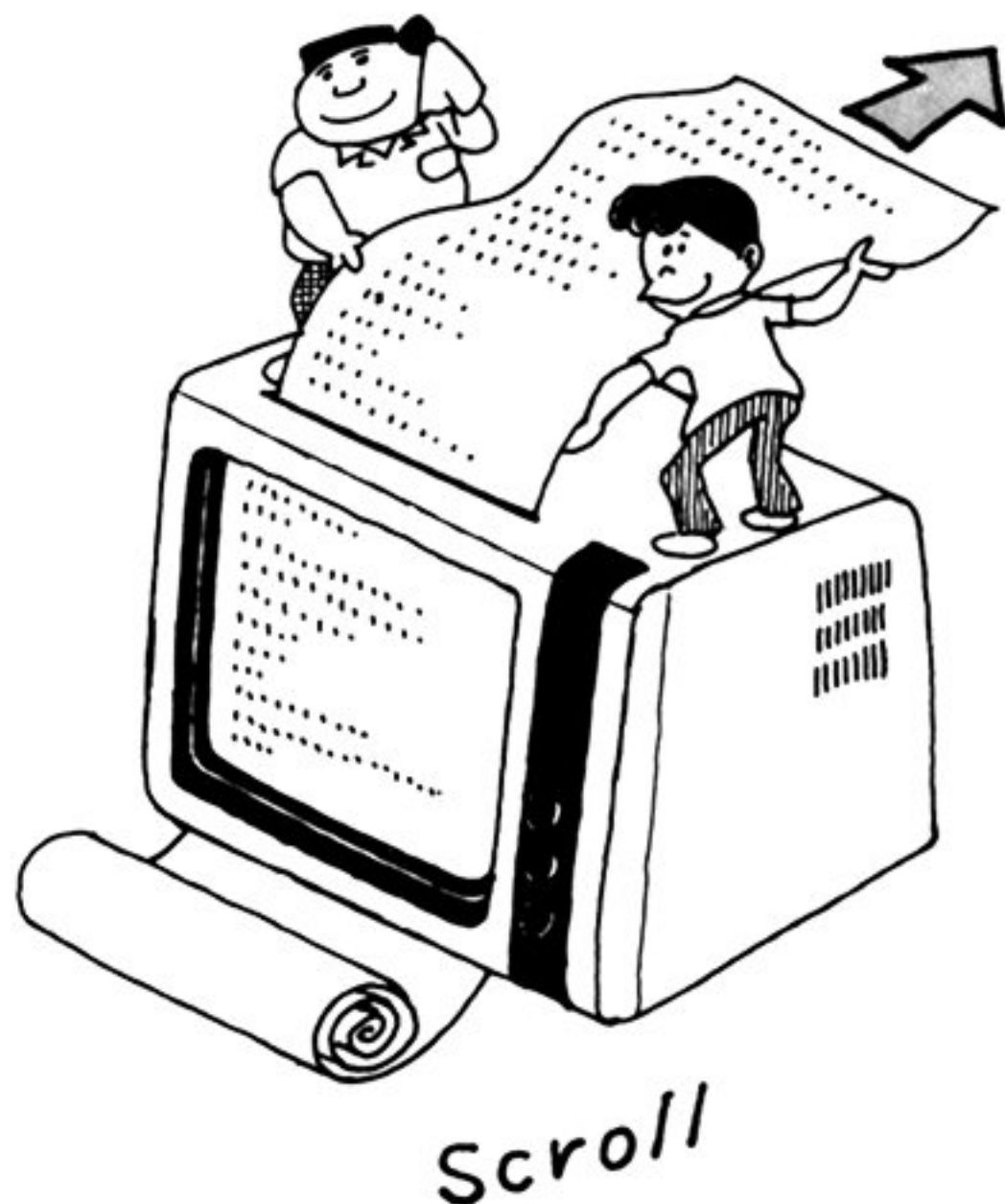
◈ **スクロール開始行**

スクロールし始める行の指定です．たとえば5と指定すると，上から6行目からスクロールを開始します．

◈ **スクロール行数**

何行分スクロールするかを示します．たとえばスクロール開始行を5と指定すると，スクロール行数は全行数から5を引いた値でなくてはなりません．

◈ **ファンクションキー表示スイッチ**

指定するときは1または0です．1を指定すると，ファンクションキーに設定してある内容が一番下の2行に表示されます．ただし，この場合はスクロールする行数が少なくなるため，たとえばWIDTH 40, 20であったら，これから2行分減らして，CONSOLE 0, 18, 1, 0またはCONSOLE 0, 17, 1, 0 にしなくてはなりません．

なお，このようにファンクションキーを表示した状態のときは，CLS キーを押したり，CLS命令を実行したりしてもファンクションキーの表示は消えません．便利な機能ですね．

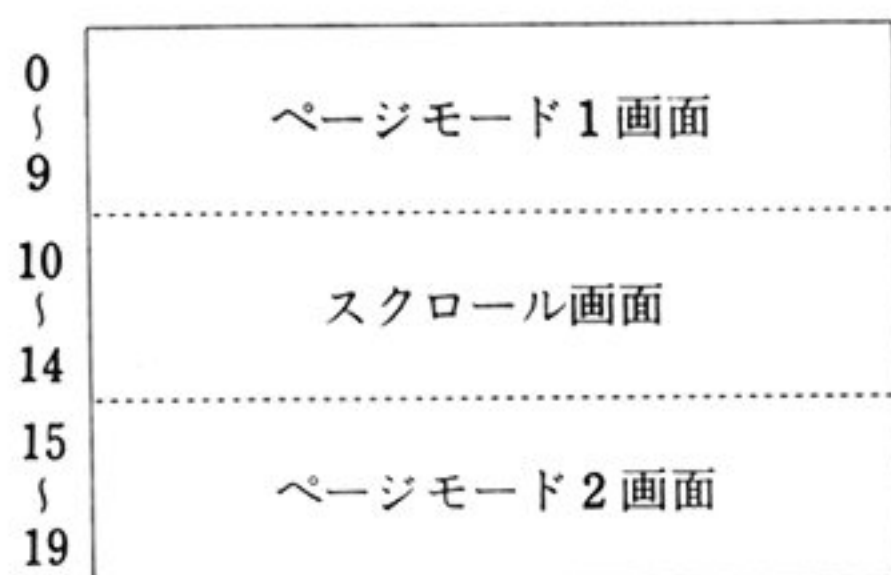
◈ **コンソールカラースイッチ**

0か1を指定します．0にすると，後述のCOLOR文で指定する色で表示されますし，1にすると緑1色で表示されるようになります．

なお，初期設定は，CONSOLE 0, 20, 0, 0になっています．ただし，FM−11の場合は，コンソールカラースイッチはなく，CONSOLE 0, 25, 0に初期設定されています．

では簡単なプログラム例を調べてみましょう．実行した結果はつぎのページに示します．

```
10 CLS
20 WIDTH 40,20
30 CONSOLE 0,17,1,0
40 PRINT"ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ"
50 PRINT
60 FOR I=1 TO 150
70 PRINT I;
80 NEXT I
```

◆ CONSOLE命令を実行すると，一般にCRT画面が三つの部分に分かれます．たとえばCONSOLE 10，5，0，0では図のようになります．

つまり，10〜14行がスクロールする部分で，カーソル移動キーを動かしても，この範囲しか動きません．ところで，CLSは画面の全部を消してカーソルがホームポジションに移動する命令でしたね．これを使うと別のページモードの画面にうつすことができるのです．つまり

CLS〔消去範囲コード〕

消去範囲コードは0から3（FM−11は5）まで指定でき，つぎのような仕事をします．

0……全画面をクリアする．
1……スクロール画面をクリアする．
2……ページモード1画面をクリアする．
3……ページモード2画面をクリアする．

このような仕事をした後，カーソルは今クリアした画面（0のときは，スクロール画面）の一番左上に移動します．なお，このコードを省略すると，0とみなされます．ます．また，FM−11の場合には，少し別の働きをしますが，その詳細については文法書などを参照してください．

```
 33   34   35   36   37   38   39   40   41   42
 43   44   45   46   47   48   49   50   51   52
 53   54   55   56   57   58   59   60   61   62
 63   64   65   66   67   68   69   70   71   72
 73   74   75   76   77   78   79   80   81   82
 83   84   85   86   87   88   89   90   91   92
 93   94   95   96   97   98   99   100   101
 102   103   104   105   106   107   108   109
 110   111   112   113   114   115   116   117
 118   119   120   121   122   123   124   125
 126   127   128   129   130   131   132   133
 134   135   136   137   138   139   140   141
 142   143   144   145   146   147   148   149
 150
Ready
```

3，17，1，0ではエラーになるわ．
行数が足りなくなるからさ．
3，14，1，0ならOKよ．
WIDTHを40，25にすればいいぞ．
WIDTHの行数指定とCONSOLEの行数指定が違うとおもしろいスクロールの仕方をするね．

```
10 CONSOLE 5,5
20 FOR I=1 TO 3:PRINT"AAAAA":NEXT:CLS 2
30 FOR I=1 TO 3:PRINT"BBBBB":NEXT:CLS 3
40 FOR I=1 TO 3:PRINT"CCCCC":NEXT
```

左のプログラムを実行してごらん…

3.3 COLOR

◆COLOR

カラーCRT上にカラーで文字や図形を表示させる方法です．カラーを表示させるには，カラーコードという色を数値で表す方法が使われます．

カラーコード表

コード	色	コード	色
0(8)	黒	4(12)	緑
1(9)	青	5(13)	水色
2(10)	赤	6(14)	黄
3(11)	紫	7(15)	白

FMシリーズの中でも機種によってその使い方に多少の相違点がありますが，ここではFM－7を中心に説明を進めます．

ところで，色の指定には，COLORという命令が使われます．

COLOR ［フォアグランドカラー］
［,バックグラウンドカラー］

むずかしい用語がでてきましたが，ここでは初めのフォアグランドカラーというのが，CRT上に写しだす文字や図形の色，そしてつぎのバックグラウンドカラーというのが，文字や図形以外の背景の色と考えておいてください．ただし，バックグラウンドカラーを変更するときは，COLORによる命令の後に，CLSの命令があったとき実行されます．

たとえば

COLOR 2 RETURN

と入力してやると，以後の文字は赤で表示されるようになります．また

COLOR 2，4：CLS RETURN

によって，以後の表示は緑の背景に赤の文字を表示するようになります．

なお，BASICを起動した初期の状態では

COLOR 7，0

になっています．ではこの初期の状態に直してからつぎのプログラムを実行してみましょう．

```
10 COLOR 1,7
20 PRINT"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ"
30 STOP
40 CLS
50 PRINT"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ"
```

これを実行すると，青で〝パーソナル　コンピュータ〞と表示し30行でストップします．そこでCONT RETURN と入れると実行を再開し，今度は白の背景色に青の文字を表示します．

CONTという命令は，STOPやENDの命令によって実行を停止したとき，そのプログラムの続きを実行せよという命令でしたね．

これによって，行番号40のCLSで背景色が変わることがわかります．

ところで，前述のカラーコード表には，(　　)の中にカラーコードに8を加えた数値が示してあります．この数値を指定すると文字の表示色と背景色が逆になって表示されます．たとえば

COLOR 10

で赤の背景色に黒の文字が表示されますし

COLOR 14，2：CLS

では，黄の背景色の中に赤の文字があらわれるわけです．では，もうひとつプログラムの例を調べてみましょう．初めには，背景色が黒でさまざまな色で〝パーソナル　コンピュータ〞と表示，

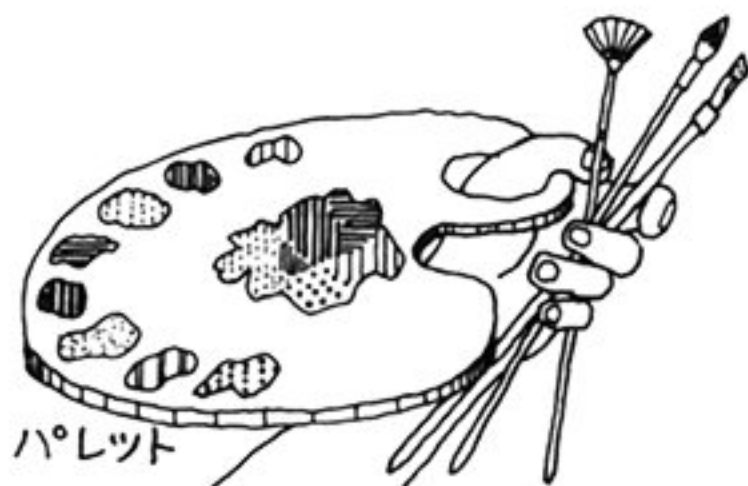
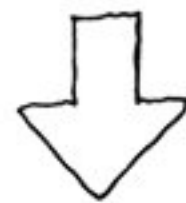
続いて背景色の異なる黒の文字を表示してくるはずです．なお，行番号60と110は，コンピュータにわざと処理時間を遅らせるための命令です．

```
10 COLOR 7,0:CLS
20 FOR I=1 TO 500:NEXT I
30 FOR J=1 TO 7
40 COLOR J
50 PRINT"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ"
60 FOR I=1 TO 500:NEXT I
70 NEXT J
80 FOR J=1 TO 7
90 COLOR J+8
100 PRINT"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ"
110 FOR I=1 TO 500:NEXT I
120 NEXT J
```

以上のプログラムは，FM－8やFM－11でもFM－7と同様な動作をします．しかし，正確にいうとこれはFM－7用の命令であり，FM－8や11は結果的には同じ形式になるため，上記のプログラムで同様な動作をする，といった方が正しいのです．その辺の詳細については，F-BASIC文法書などを参照してください．

COLOR文には，つぎのようなものもあります．この命令自身は，FM－7およびFM－11に通用するのですが，FM－11は前述の命令の使い方がFM－7とは少し異なるので，ここで示したプログラムではうまく動作をしてくれません．FM－11の場合については，後で触れることにします．なお，FM－8には適用できません．

COLOR ＝（パレットコード，カラーコード）

パレットとは絵具のとく道具のことですね．前述のCOLOR文で指定した，フォアグランドカラーやバックグラウンドカラーを瞬時に変えてしまう便利な命令です．とにかくプログラム例で調べてみましょう．

COLOR A，B

COLOR(A，X)

パレットコード　カラーコード

Bの色を変えることもできる．

COLOR(B，X)

ただし，パレットコードが0のときは，黒以外の色にすることはできない．

```
10 COLOR 4,0:CLS
20 PRINT"FMｼﾘｰｽﾞ""
30 STOP
40 COLOR=(4,2)
50 STOP
60 COLOR=(4,4)
70 COLOR 7,0:CLS
```

行番号10で"FMシリーズ"というPRINT文の文字の色を緑に指定していますね．つぎに，行番号40でパレットコードに当たる4を2に直しています．そこで行番号30でストップした後，CONTと入れると，色がたちまち赤に変わって行番号50でまた停止します．

ふたたびCONTと入れると行番号60でパレットコードを，もとの4に戻しています．この処理をしておかないと，つぎに別の仕事をさせようとするとき，COLOR　4で赤になってカラーコードと一致しないため困ってしまうからです．

つまり，BASICでは初期の状態では，パレットコードはカラーコードの0～7と一致しています．したがって，このCOLOR文を使わなければ，パレットコードはカラーコードと全く同じように使用することができるのです．

別のプログラム例によって，さらにこのCOLOR文の使い方を調べてみましょう．

```
10 COLOR 2
20 CLS
30 LOCATE 13,10
40 PRINT"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ"
50 FOR I=1 TO 3000:NEXT I
60 COLOR=(2,7)
70 FOR I=1 TO 3000:NEXT I
80 COLOR=(2,6)
90 FOR I=1 TO 3000:NEXT I
100 COLOR=(2,5)
110 FOR I=1 TO 3000:NEXT I
120 COLOR=(2,2)
130 FOR I=1 TO 3000:NEXT I
140 COLOR 7:CLS
```

このプログラムを実行すると，画面中央に表示されている〝パーソナル　コンピュータ〟の文字がネオンサインのように色を変えていきます．すなわち，行番号10で示したフォアグランドカラーである2を，行番号60，80，120のパレットコードで，いろいろな種類のカラーコードに割りつけているために，それにしたがって，つぎつぎに色が変わっているわけです．

今度は，文字だけではなく背景色もこの命令文を使っていろいろと変化させてみましょう．

```
10 COLOR 1,2:CLS
20 LOCATE 13,9
30 PRINT"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ"
40 COLOR 3
50 LOCATE 16,11
60 PRINT"FMｰｼﾘｰｽﾞ"
70 FOR I=1 TO 3000:NEXT I
80 COLOR=(2,4)
90 FOR I=1 TO 3000:NEXT I
100 COLOR=(1,0)
110 FOR I=1 TO 3000:NEXT I
120 COLOR=(1,2):COLOR=(3,1)
130 FOR I=1 TO 3000:NEXT I
140 COLOR=(1,7):COLOR=(2,1)
      :COLOR=(3,2)
150 FOR I=1 TO 3000:NEXT I
160 COLOR 7,0:CLS
170 FOR I=1 TO 7
180 COLOR=(I,I)
190 NEXT I
```

このプログラムでは，初めの段階では，上側の文字が青，下側の文字が紫，そして背景色が赤の状態から始まります．そして，少しずつ時間をおいて各文字や背景の色が切り替わっていくのが，分かりますね．このように，背景のような大きな面積でも，その色を瞬時に切り替えることができるのが，パレットコードを使ったこの命令のおもしろいところです．

なお，行番号170～190は，パレットコードとカラーコードとが1対1の対応をしなくなったときに，初期の状態にもどしてやるためのプログラムです．もっとも，ここに示したプログラムの例では，すべてのコードが対応しなくなったわけではありませんが，このような処理しておけば，どのコードが変わってしまったか，などといちいち考える必要がありません．

ここで，FM－11の場合のCOLOR文について触れておきます．なお，詳細については文法書を参照してください．

> **COLOR** [**テキストカラー**]
> 　　　[，[**バックグラウンドカラー**]
> 　　　[，[**フォアグラウンドカラー**]
> 　　　　[，**アトリビュート**]]]

FM－7の場合よりも，指定項目が多くなっていますね．はじめのテキストカラーというのは，表示する文字の色を設定するところです．ただしこれはパレットコードではなくカラーコードなので，パレットを変化させても，文字の色を変えることはできません．

つぎのバックグラウンドカラーとフォアグラウンドカラーについては，FM－7の場合と同じ扱いです．したがって，パレットコードにより色を変えることができます．さき程のプログラム例はFM－7の場合の作り方をしているので，FM－11ではうまく動いてくれないのです．命令の指定の方法を少し変更してやれば，正常に動いてくれますね．なお，この本でこれ以降にのべるカラーに関する説明でも同様なことがいえます．たとえば，FM－11でテキストカラーやバックグラウンドカラーの指定を省略し，フォアグランドカラーだけを指定したいときには

COLOR ，，X

のように，〔　〕付きの指定がある場合，不必

要なコードは省略してもよいのですが，コンマだけは残さなくてはなりません．つぎのアトリビュートだけを指示したいときは

COLOR ，，，Y

でよいわけです．

最後のアトリビュートというのは，画面に表示している文字に，いろいろな機能をあたえる部分で，0から7までの数字であらわします．

0………通常の表示
1………反転表示
2………点滅表示
3………反転点滅表示
4………高輝度表示
5………高輝度反転表示
6………高輝度点滅表示
7………高輝度反転点滅表示

◈SCREEN

FM－7および11にはSCREENという命令もあります．色は青，赤，緑の3原色でできています．そして，CRT画面のすべての点は，ビデオRAM（VRAM）というメモリに記憶されています．そこで，このVRAMに対して直接動作を指示してやるための命令です．FM－7では，つぎの命令を使うことができます．

SCREEN［アクティブVRAMコード］
［，ディスプレイVRAMコード］

VRAMの中では，色の違いは3ビットに対応しています．つまり，下の表のようになります．

色のコード	ビット2 G	ビット1 R	ビット0 B	色
0	0	0	0	黒
1	0	0	1	青
2	0	1	0	赤
3	0	1	1	紫
4	1	0	0	緑
5	1	0	1	水色
6	1	1	0	黄
7	1	1	1	白

さて，ここで出てきたSCREENという命令の中で，アクティブVRAMコードというのは，この部分で色のコードを指定すると，それ以降はそのビット以外には，書き込みができなくなることをあらわしているのです．

たとえば，COLOR　7の状態で

SCREEN　2：PRINT″A″

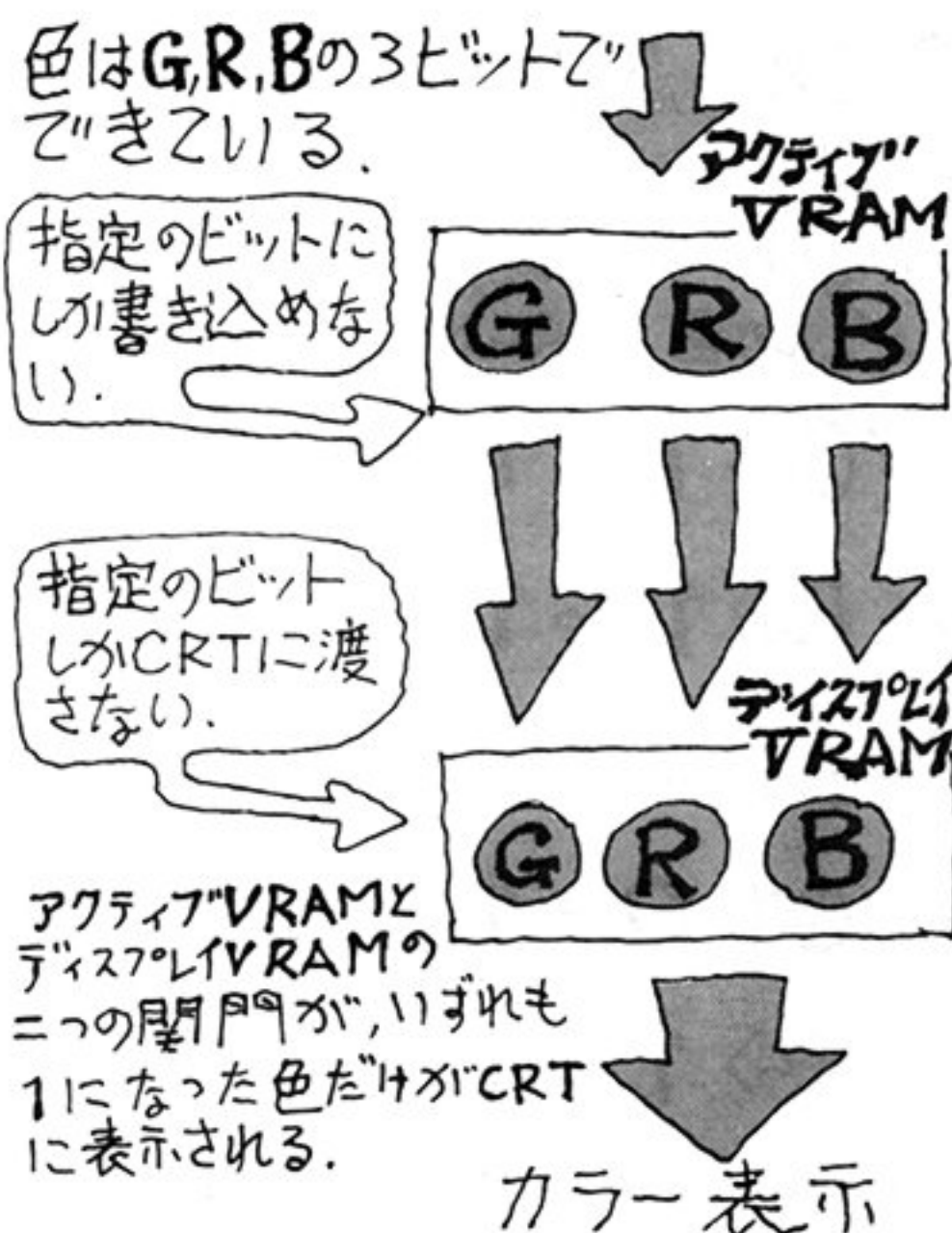
と入れると，2はR（赤）以外のビットに書き込むことができなくなる状態ですから，赤でAと表示してきます．

つぎのディスプレイVRAMコードというのは，以降ここで指定したビットだけを，CRT画面に表示する働きをします．

たとえば，COLOR 7の状態で

SCREEN　6，5：PRINT″A″

とした場合を考えてみましょう．アクティブVRAMコードは6ですから，G（緑）とR（赤）のビットは1であり，このままの状態であれば黄の文字がでるはずです．ところが，ディスプレイVRAMコードの方はGとB（青）が1になっているため，実際に画面に表示される文字はG（緑）になるはずです．

なお，初期の段階ではSCREEN　7，7の状態になっています．上記のようにSCREENをいろいろと変えたあとで，元の状態に戻したいときは，PF 9を押すとSCREEN 7，7にもどります．

では，ここで簡単なプログラムの例を調べてみましょう．

```
10 COLOR 6,3:CLS
20 LOCATE 13,10
30 PRINT"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ"
40 FOR I=1 TO 5000:NEXT I
50 SCREEN ,4
60 FOR I=1 TO 5000:NEXT I
70 SCREEN ,1
80 FOR I=1 TO 5000:NEXT I
90 SCREEN 3,3
100 COLOR 7,1:CLS
110 LOCATE 13,10
120 PRINT"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ"
130 FOR I=1 TO 5000:NEXT I
140 SCREEN 7,7
150 COLOR 7,0:CLS
```

初めに紫の背景の中に黄の文字が，続いて黒の背景の中に緑の文字が，つぎに青の背景の中に黒の文字が，そして最後に青の背景の中に紫の文字が表示されてきますね．なお，行番号140，150は初期の状態にもどすためのプログラムです．

同じSCREEN文も，FM−11の場合は画面の解像度がよく（640×200または640×400ドット）また内蔵しているグラフィックVRAMの数も多い（192KB）などのため，さらに複雑な機能を持っています．ここでは，簡単にその概要を示すことにします．詳細は文法書を参照してください．

SCREEN［画面モード］［,［アクティブページ］［,［ディスプレイページ］［,［アクティブバンクコード］［,ディスプレイバンクコード］［,［ビデオ出力コード］］］］］

画面モードは，グラフィック画面の解像度やカラー表示，単色表示，スクロールモードを設定するところです．

アクティブページは，VRAMのどのページに読み書きするかを設定するところです．

ディスプレイページは，VRAMのどのページを画面に表示するかを設定するところです．

アクティブバンクコードは，単色モードとグラフィックスクロールモードのときに，三つあるバンクのうちの，どのバンクが読み書き可能かを指定するところです．

ディスプレイバンクコードは，単色モードとグラフィックスクロールモードのとき，三つあるバンクのうちの，どのバンクを画面に表示するかを指定するところです．

ビデオ出力コードは，カラーCRTとグリーンCRTへ，それぞれグラフィック画面とテキスト画面を，どのような組み合わせで出力するかを設定するところです．

3.4 PSET

いよいよ，CRT 上に細かく点や線を描いたり，色を塗ったりする命令の作り方です．

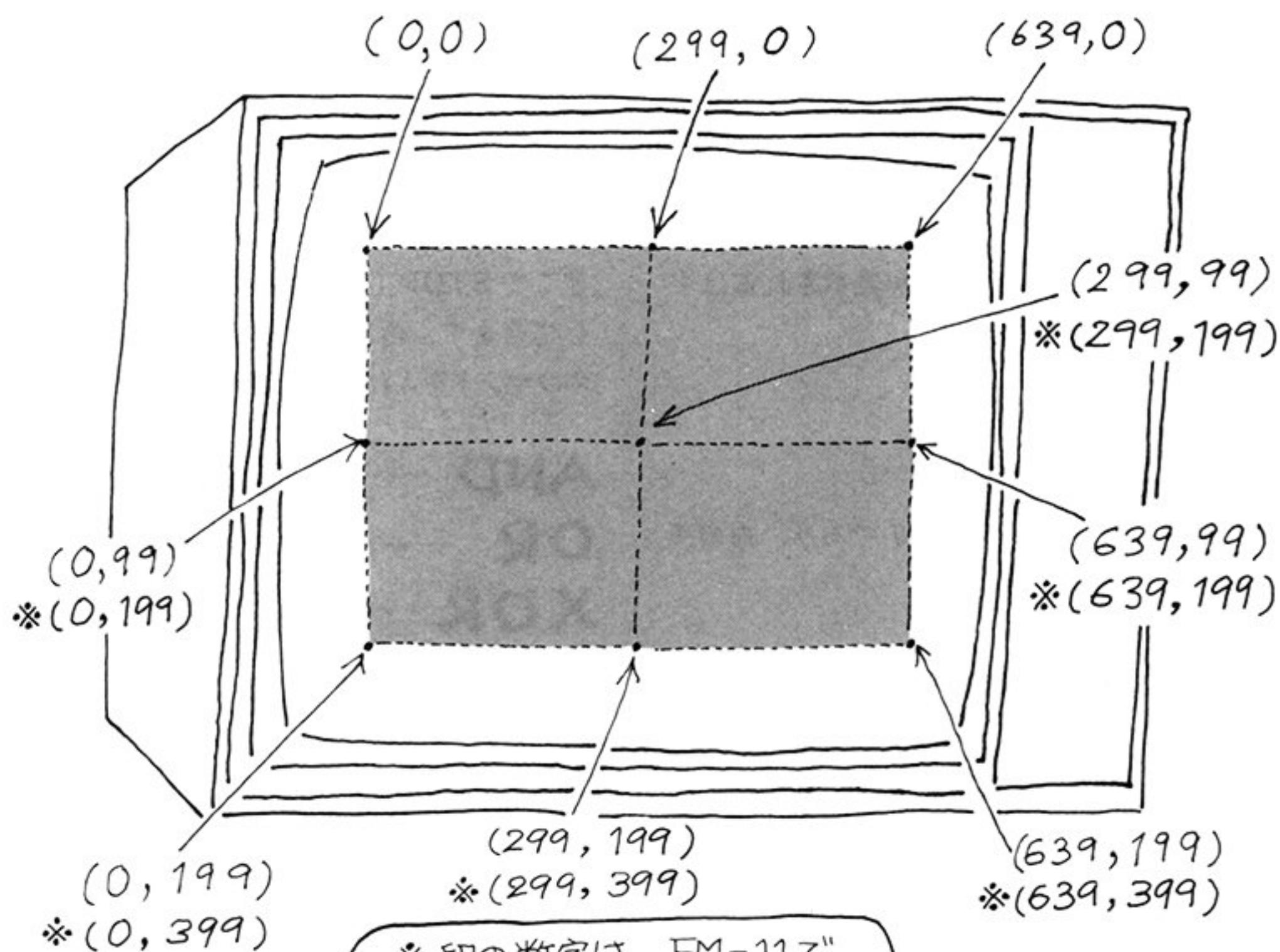
まず知っていただきたいことは，文字や記号を表示するときも，たとえばLOCATE 文で座標上のどの位置かを指定したように，CRT 上の全ての点が座標によって指定できるということです．

CLS キーを押して，画面をきれいにしたあと，PSET（300，100，4）RETURN と実行してみてください．CRT のほぼ中央付近に，ポツンと緑の点が写し出されます．この位置は，( ) の中の初めの二つの数300，100によって，指定された位置なのです．次の4はパレットコードであることはすぐわかりますね．

なお，FM－7 および FM－8のCRTは，縦640，横200 の点の集まりで表示されますが，FM－11では 640×400 です．ここでは 640×200 の例を中心に話を進めているため，FM－11に適用するときは，このことに注意してください．

PSET （水平位置，垂直位置
[，[パレットコード]　[，機能]]）

ここで水平位置，垂直位置を表示するドットの座標は，必ずしも整数型でなくてもよく，実数型の場合は小数点第1位の桁が四捨五入されて，整数になります．たとえば（12.345，67.89）なら（12，68）になるわけです．

[ ]の部分は場合により省略してもかまわないという意味であることは，今までの命令に対する説明と同じですが，ここでパレットコードを省略すると，直前までのCOLOR 文で使われていた指定色になることはいうまでもありません．

次は，三角関数（**2.16** 参照）の **sin**，**cos** の曲線を描かせるプログラムです．白線で **sin** 曲線を，青線で **cos** 曲線を描きます．

```
10 CLS:COLOR 7,0
20 PRINT "SIN(A),COS(A)"
30 PI=3.14159:D=PI/180
40 FOR I=0 TO 360
50 X=100+I
60 Y=100-SIN(D*I)*50
70 PSET(X,Y)
80 Z=100-COS(D*I)*50
90 PSET(X,Z,1)
100 NEXT I
```

PSET の最後に記入する「機能」には，論理式である AND，OR，XOR のいずれかが指定可能です（論理式の説明は後述します）．

たとえば OR とすると，現在画面に表示しているドットのパレットコード（３桁の２進数）と，この命令文で指定したパレットコードとの論理和の結果得られたカラーが表示されることになります．

仮に，現在表示している色を緑とします．緑のコードは４，つまり２進数でいうと 100 です．そこで，PSET（310，100，３，OR）とすると，コード３は紫ですから，紫の点が表示されるはずですが，OR があるため，

緑	4 ──→	100
紫	3 ──→	011
論理和	──→	111

の結果，コードは 111，つまり７であり，白の点が出てくるわけです．

この一連の動きを実行してみると，次のプログラムになります．ただし，このプログラムでは，動きがよくわかるようにするため，最終的に三つの点を表示するようにしています．CRT 上の色の変化をよく観察してください．

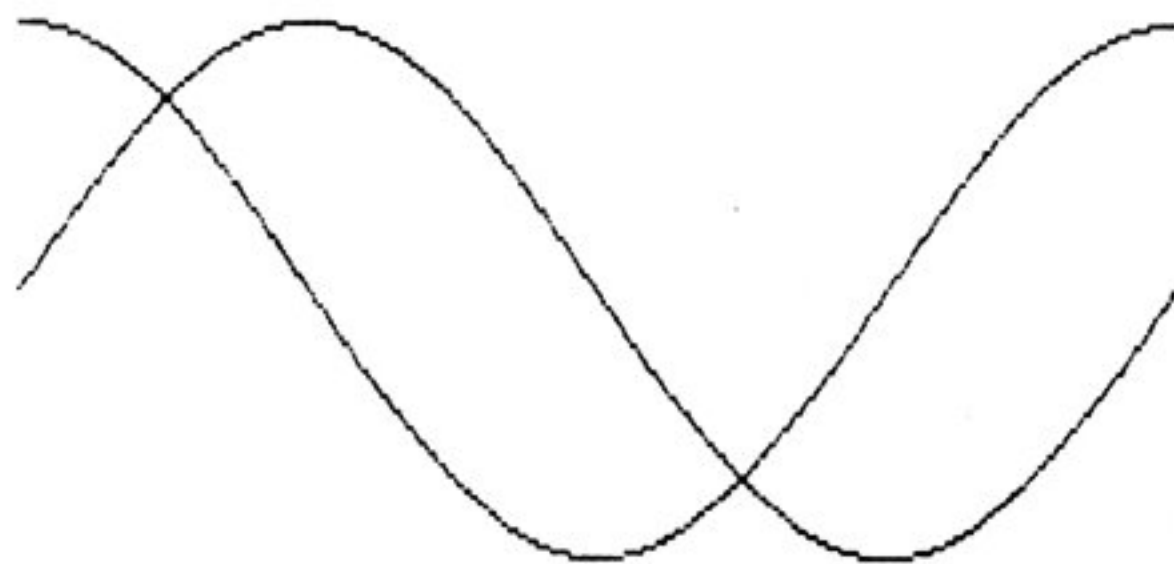
もし，OR のかわりに AND，つまり論理積にすると，

白	7 ──→	111
紫	3 ──→	011
論理積	──→	011

011 は３ですから，紫色の点が出てきます．

```
10 COLOR 7,0
20 CLS
30 PSET(290,100)
40 PSET(310,100,4)
50 STOP
60 PSET(290,100,1,AND)
70 STOP
80 PSET(310,100,3,OR)
90 STOP
100 PSET(310,100,3,AND)
110 PSET(300,100,5,OR)
120 END
```

ここで STOP の命令は，プログラムを一時停止する機能を持っています．一時停止した上で，どこで STOP したかわかるように，行番号を表示してきます．再び実行を続けさせるには，CONT のコマンドを入れてください．実行を再開します．

AND ……… 論理積
OR ……… 論理和
XOR ……… 排他的論理和

SIN(A),COS(A)

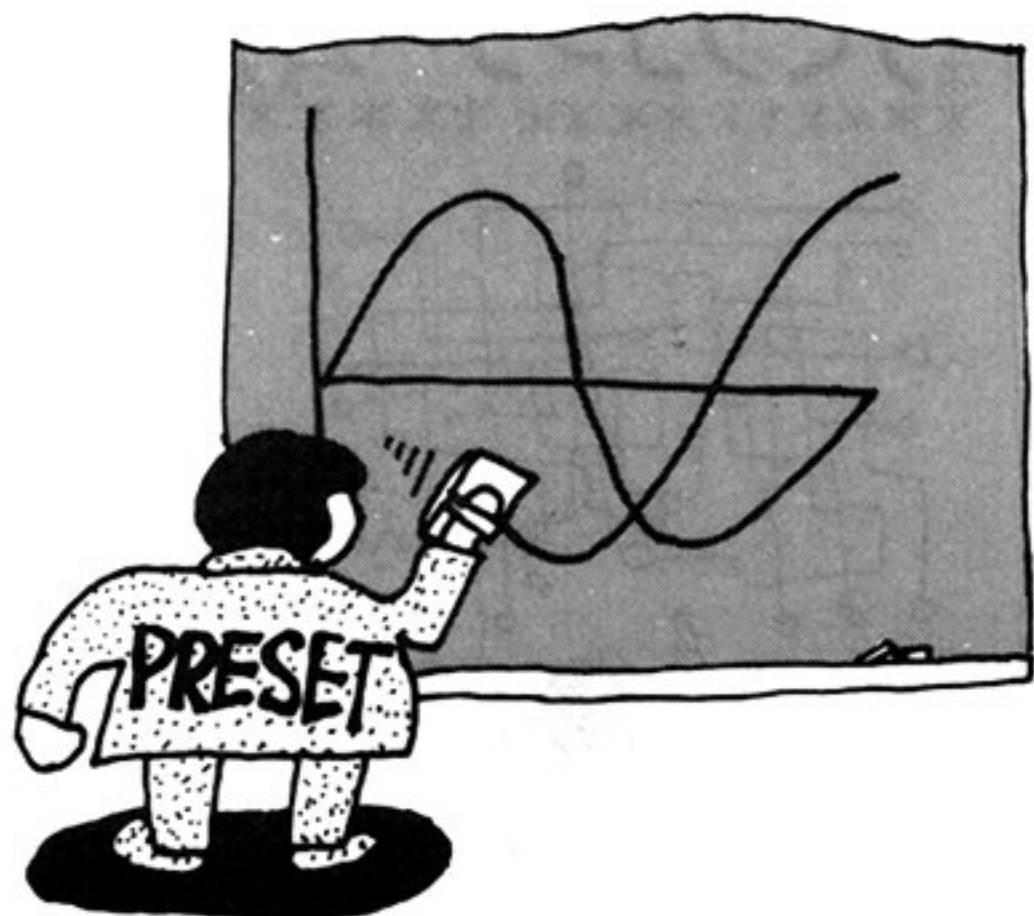
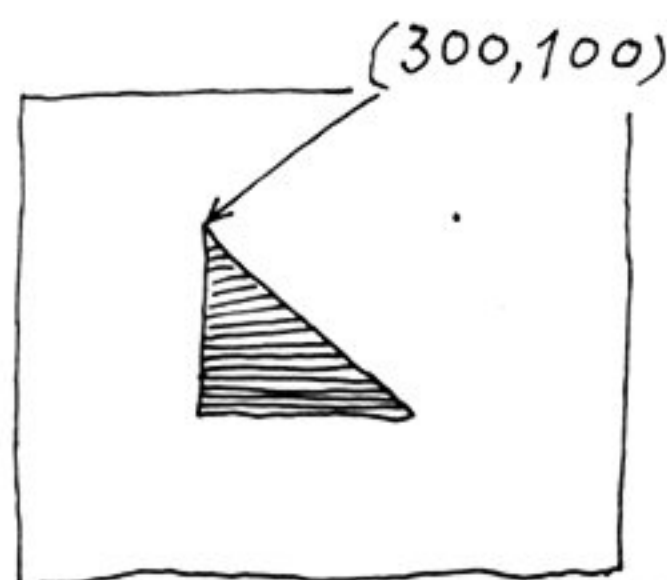
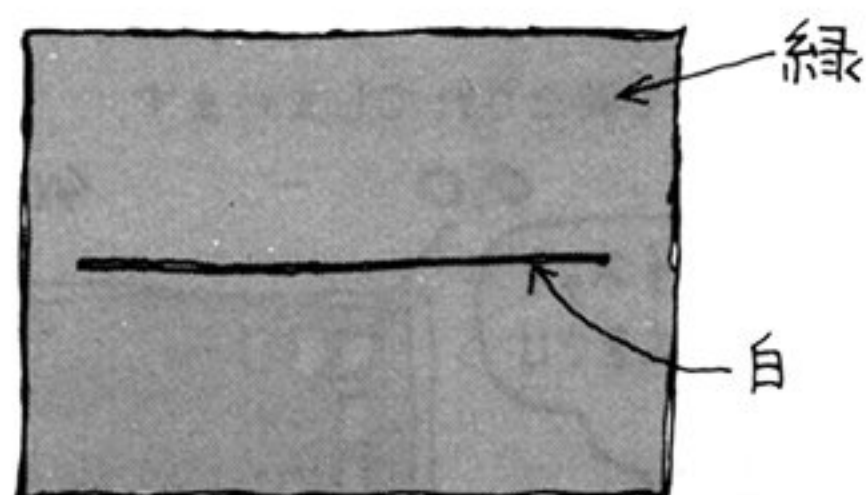
3.5 PRESET

この命令は，点の色を背景色と同じ色にしてしまいます．使い方によっては，線を消してしまう，と考えることができます．

PRESET （水平位置，垂直位置）

PSETで描いたsin，cos曲線をPRESETで消してしまうプログラムを作ってみましょう．

行番号110以後が新しく追加したPRESETで，曲線を消してしまう命令です．

```
10 CLS
20 PRINT"SIN(A),COS(A)"
30 PI=3.14159:D=PI/180
40 FOR I=0 TO 360
50 X=100+I
60 Y=100-SIN(D*I)*50
70 PSET(X,Y)
80 Z=100-COS(D*I)*50
90 PSET(X,Z,1)
100 NEXT I
110 FOR I=0 TO 360
120 X=100+I
130 Y=100-SIN(D*I)*50
140 PRESET(X,Y)
150 Z=100-COS(D*I)*50
160 PRESET(X,Z)
170 NEXT I
```

```
10 CLS
20 FOR I=1 TO 8
30 FOR J=0 TO 20
40 FOR K=300 TO 300+3*J
50 PSET(K,J+100,I)
60 NEXT K,J
70 FOR J=0 TO 20
80 FOR K=300 TO 300+3*J
90 PRESET(K,J+100)
100 NEXT K,J,I
```

```
10 COLOR,4
20 CLS
30 FOR I=0 TO 440
40 PSET(100+I,100,7)
50 NEXT I
60 FOR J=0 TO 1000:NEXT J
70 FOR I=0 TO 440
80 PRESET(100+I,100)
90 NEXT I
```

行番号40で白線を描いたあと
行番号80で消す．

3.6 LINE

CRT 上に線や箱を描くための LINE の命令には，文字や記号のキャラクタを使う方法と，ドットによる方法とがあります．まず，キャラクタによる方法を示します．

LINE [@][(X_1, Y_1)]−(X_2, Y_2), **文字列** [,[**パレットコード**] [, {B / BF}]]

@はアットマークといいます．

X_1，Y_1 および X_2，Y_2 は図のように CRT 上の座標を示します．もし X_1，Y_1 を省略したときは，直前に実行された LINE 文の X_2，Y_2 が新しい LINE 文の X_1，Y_1 になります．

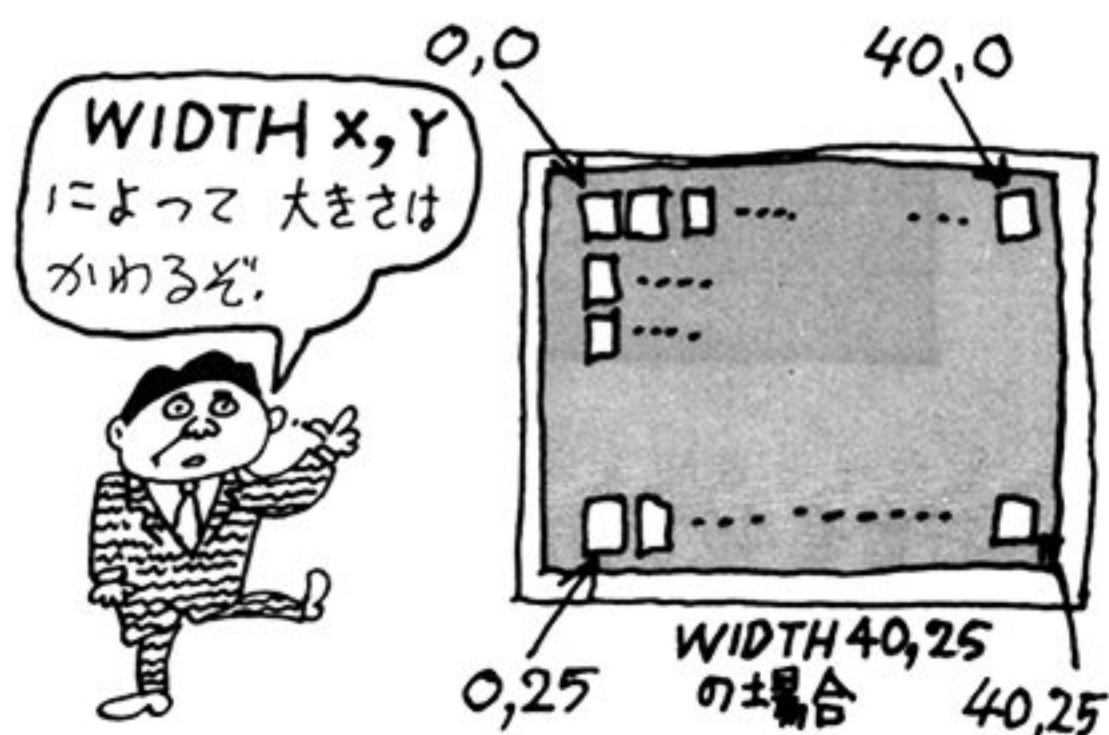
```
LINE@(0,0)-(20,20),"ABC",4
```

ABC という文字列の先頭の文字Aで，座標 0，0 から 20，20 までを直線状に結んでくれます．なお，このときの文字の色はパレットコードが 4 であるため，緑です．

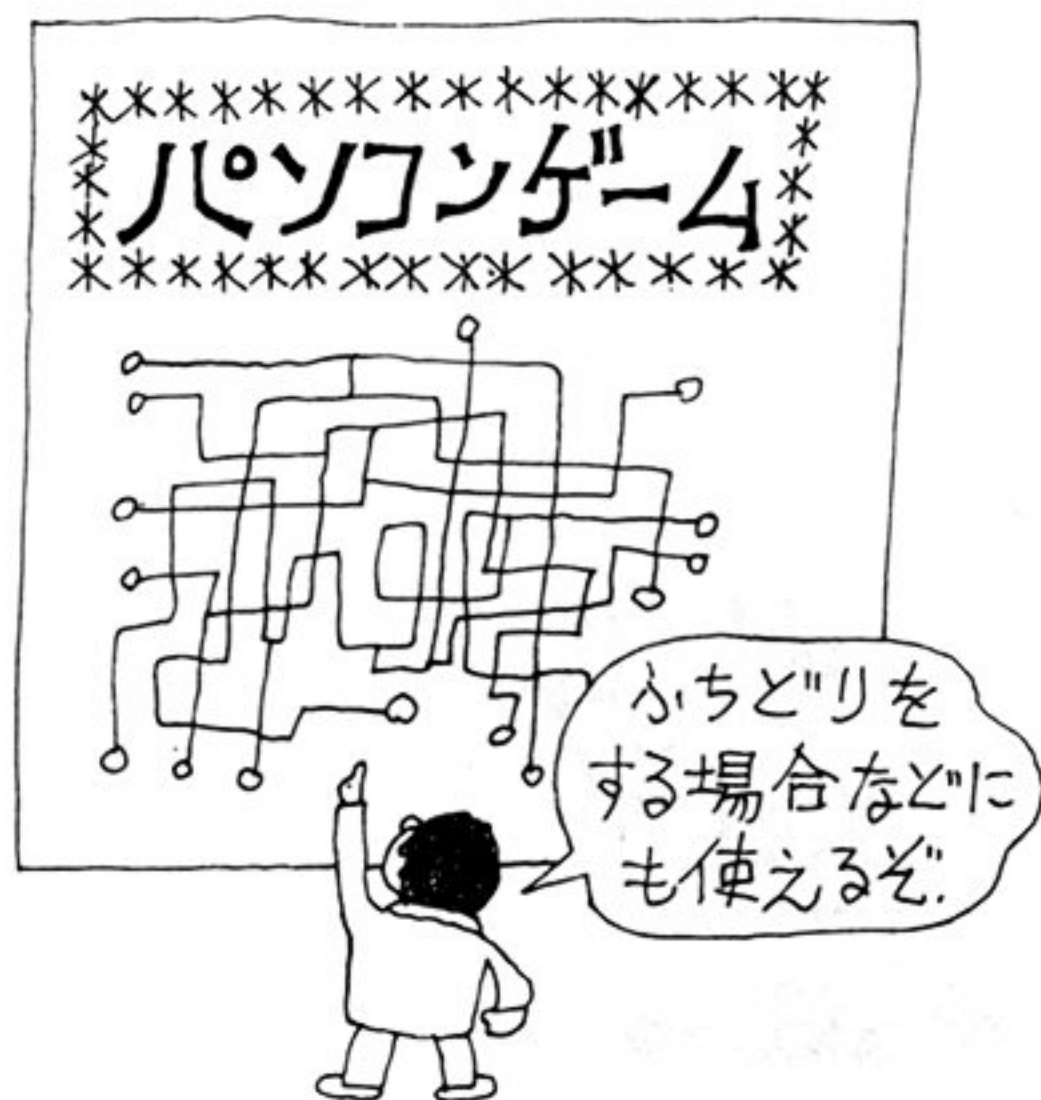
```
LINE@(0,0)-(20,20),"ABC",4,B
```

BはBoxの略です．つまり，（0，0）−（20，20）を対角線とする長方形状にAが並びます．

```
LINE@(0,0)-(20,20),"ABC",4,BF
```

BのかわりにBFにすると，長方形の内側も全てAによって満たされてしまいます．

```
10 CLS
20 A$="ABCDE"
30 LINE@(0,0)-(10,10),A$,,B
40 LOCATE 0,11
```

```
10 CLS
20 A$="ABCDE"
30 LINE@(0,0)-(10,10),A$,
     2,BF
40 LOCATE 0,11
```

赤色の文字 →

ドットを使って線や箱を描くのは，次の命令によります．

LINE [@][(X1, Y1)]−(X2, Y2), 機能[,[パレットコード][,{B / BF}]]

キャラクタの場合と異なる点は，文字列のところが機能にかわっている点です．

機能は，PSET，PRESET，AND，OR，XOR のいずれかを指定します．

```
LINE@(0,0)-(20,20),PSET,4
```

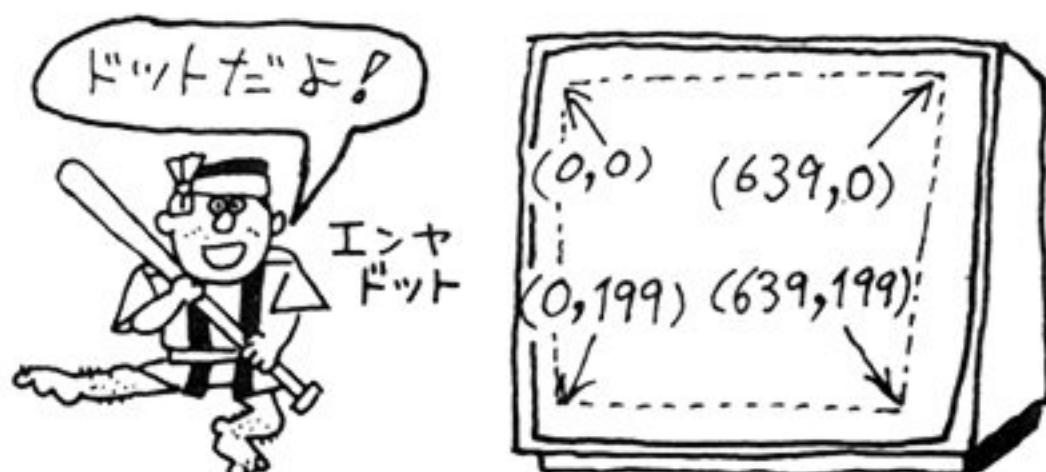
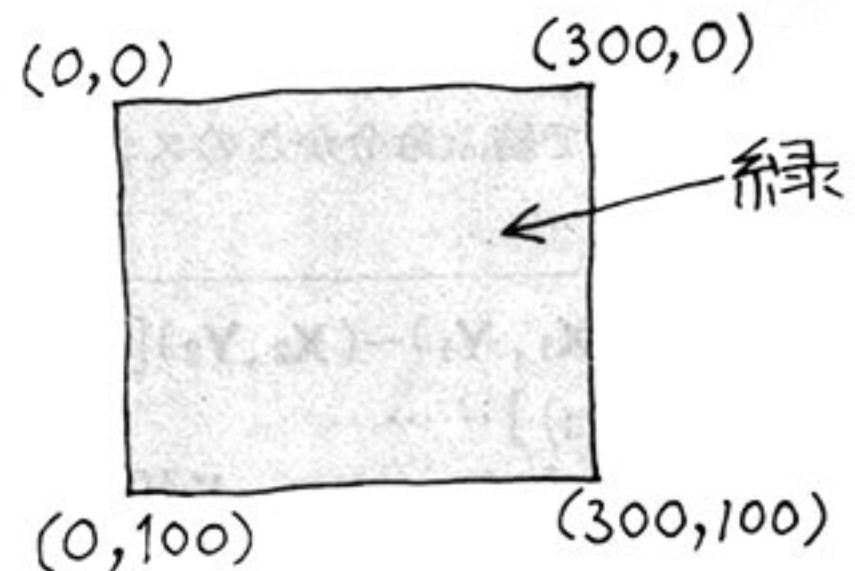
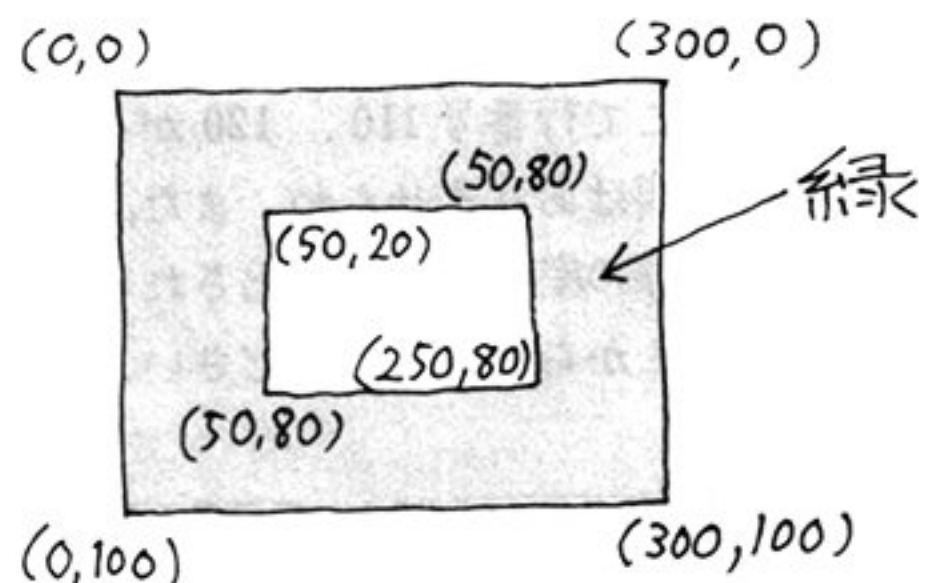
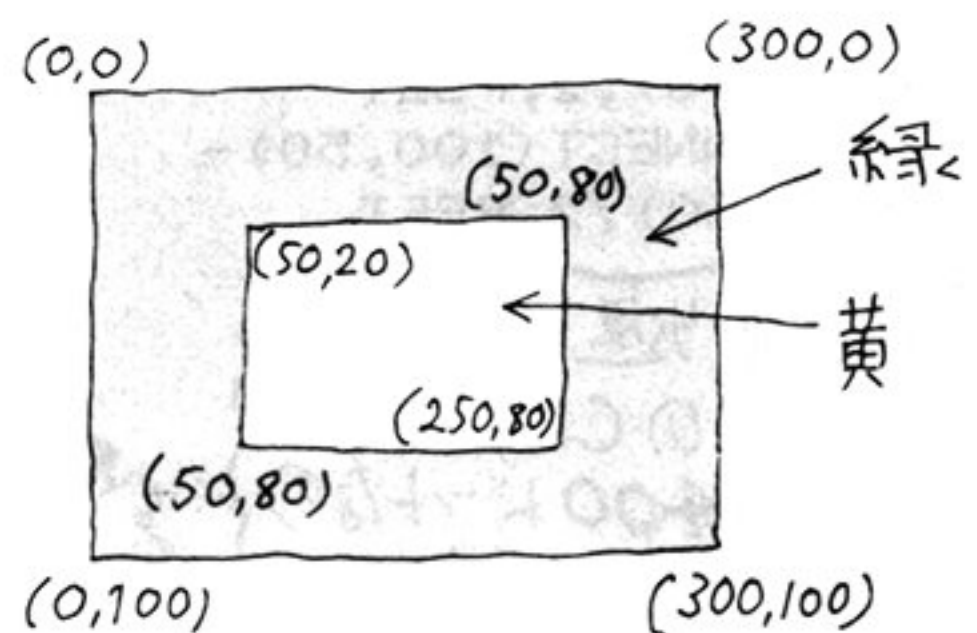
この場合の座標はキャラクタではないので，横0〜639，縦0〜199 になるわけです．0,0 〜20,20 の間を直線で結んでくれます．

```
LINE@(0,0)-(20,20),PSET,4,B
LINE@(0,0)-(20,20),PSET,4,BF
```

この意味はもうおわかりですね．注意しなくてはならないのは，もしパレットコードを省略して，BまたはBF を指定するときは，コンマを二つ並べてやらなくてはならないという点です．

```
10 CLS
20 LINE@(0,0)-(50,50),PSET,4,BF
30 COLOR 5
40 LINE@(60,60)-(110,110),PSET,,BF
50 LOCATE 0,11
```

ここで，を一つにすると，BまたはBF を変数とみなし，パレットコードが0になってしまいます．なお，パレットコード指定を省略すると，直前のCOLOR文のパレットコードで表示されます．

機能のところをAND，OR，XOR で指定したときは，指定色と現在画面に表示されているパレットコードとの論理演算を実行し，その結果のカラーで表示してきます．論理演算については，3.4PSET のところで練習しましたね．これを参照してください．

```
10 CLS
20 LINE@(0,0)-(300,100),
     PSET,4,BF
30 LOCATE 0,11
```

```
10 CLS
20 LINE@(0,0)-(300,100),
     PSET,4,BF
30 LINE@(50,20)-(250,80),
     PRESET,4,BF
40 LOCATE 0,11
```

```
10 CLS
20 LINE@(0,0)-(300,100),
     PSET,4,BF
30 LINE@(50,20)-(250,80),
     OR,2,BF
40 LOCATE 0,11
```

3.7 CONNECT

点と点とを直線で結ぶ命令がこのステートメントです．

CONNECT (X1, Y1)−(X2, Y2)[−(X3, y3)]…… [,[パレットコード][,機能]]

点がいくつもある場合には，……−(Xn, Yn)−……によってつないでいくことができます．

機能の部分はPSET，PRESET，AND，OR，XORのいずれかがくるわけです．

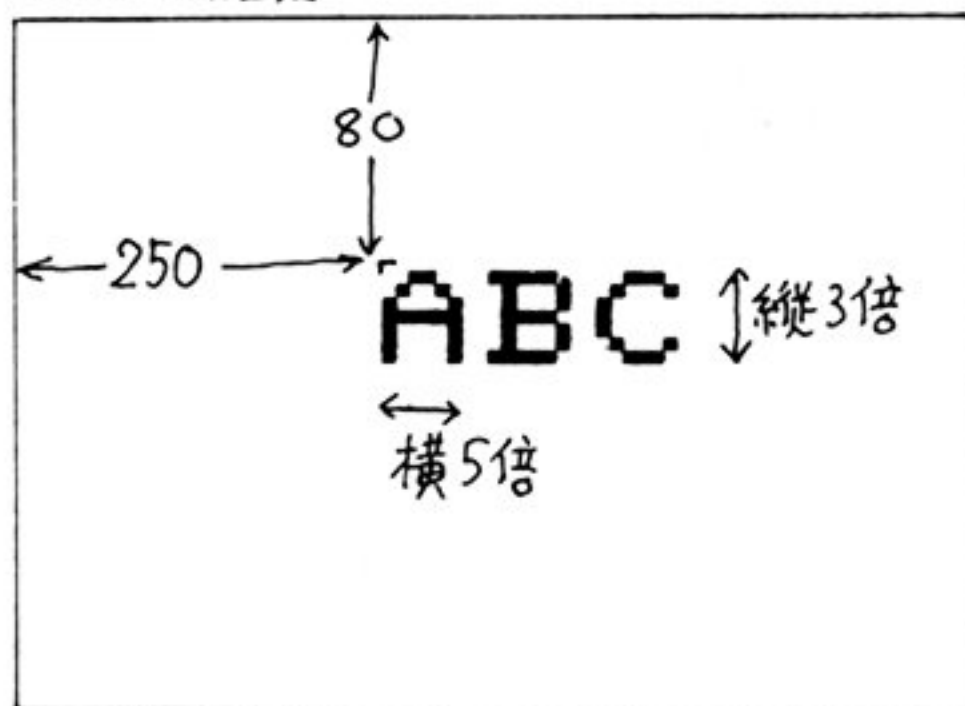
この命令には，特にむずかしい問題はありません．では**3.4 PSET**のところで示した，sin, cos曲線に座標軸（X軸，Y軸）を追加してみることにします．ここで行番号110，120がそれです．特に説明の必要はありませんね．また，この指定によって，赤線の座標軸が描き出されるのはなぜか，プログラムから検討してください．

```
10 CLS
20 PRINT"SIN(A),COS(A)"
30 PI=3.14159:D=PI/180
40 FOR I=0 TO 360
50 X=100+I
60 Y=100-SIN(D*I)*50
70 PSET(X,Y)
80 Z=100-COS(D*I)*50
90 PSET(X,Z,1)
100 NEXT I
110 CONNECT(100,100)-
(460,100),2,PSET
120 CONNECT(100,50)-
(100,150),2,PSET
```

3.8 SYMBOL

画面上にドットを使って文字を表示する命令であり，文字列の角度，大きさを指定によって変えることができます．

SYMBOL (X, Y), 文字列, 横倍率, 縦倍率[,[パレットコード][,[角度コード][,機能]]]

簡単なプログラム例を示します．

初めの（250，80）は，ドットの座標での座標指定です．図のように，文字列の表示を始める枠内の，左上のドット座標を示します．このプログラムの例では終わりから2番目の文字が0になっているため，ノーマルな位置を表示していますが，この数字を0～3と変えて，どの位置から文字列の表示が始まるか確認しておくとよいでしょう．要するに，この点からABCの文字列が書き始められるということを知っておいてください．

座標の指定の次が文字列です．ここを文字変数にして，文字列の内容を別のところで指定してやることももちろん可能です．このように，ここで示した文字や数字を変数の扱いにして，変数の中に入っている文字や数字を別のところで指定してやることは，どの場合も可能です．

次の二つが文字の横倍率，縦倍率を示す数字です．文字は，8×倍率ビットの大きさで表示されてきます．

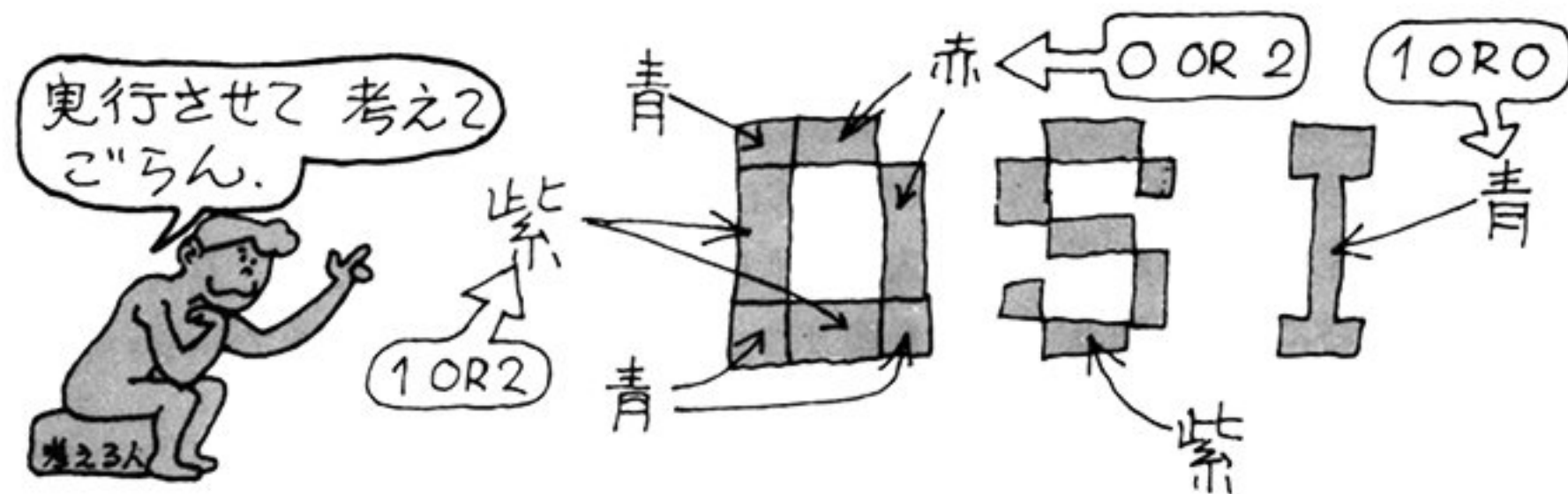
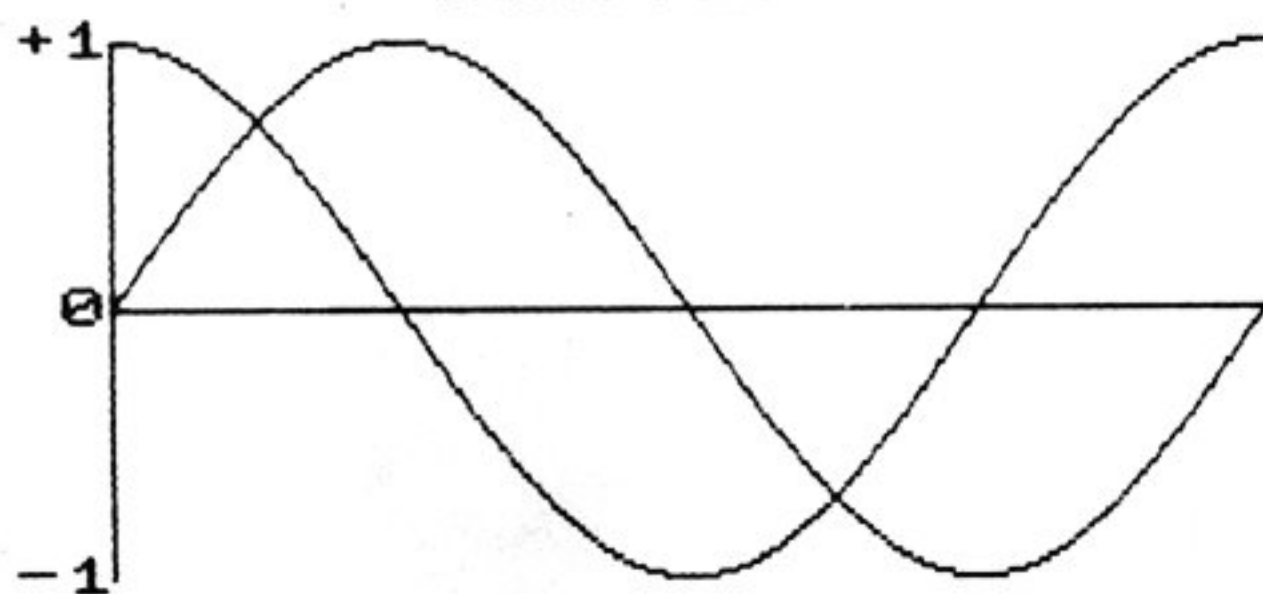
パレットコードの指定は，もう何度も出てきたので省略します．

角度コードは0が正常，1が時計と反対方向に90°，2が180°，3が270°回転した状態で表示されます．

機能はPSET，PRESET，AND，OR，XOR，NOTの指定が可能です．

```
10 CLS
20 SYMBOL(250,0),"LSI",5,3,1
25 STOP
30 SYMBOL(250,0),"OS ",5,3,2,0,OR
```

COLOR 1……青
COLOR 2……赤
COLOR 3……紫

25 で"STOP.そこでCONTと入れる.

```
10 CLS
20 SYMBOL(200,35),"Sin,Cos カーブ",2,1,4
30 D=3.14159/180
40 FOR I=0 TO 360
50 X=100+I
60 Y=100-SIN(D*I)*50
70 PSET(X,Y)
80 Z=100-COS(D*I)*50
90 PSET(X,Z,1)
100 NEXT I
110 CONNECT(100,100)-(460,100),2,PSET
120 CONNECT(100,50)-(100,150),2,PSET
130 SYMBOL(68,46),"+1",2,1,2
140 SYMBOL(83,96),"0",2,1,2
150 SYMBOL(68,146),"-1",2,1,2
```

角度や大きさを変えて出した例

ABC
ABC

左は前述のサインカーブに，CONNECT命令やSYMBOL命令を使った例だ．

3.9 CIRCLE

CRT 上のあなたの希望する位置に，あなたの希望する色，形の円を描いてみませんか？

```
CIRCLE[@](水平位置, 垂直位置),半径
        [,[パレットコード][,[比率]
        [,[開始位置][,[終了位置]
        [,[{F
            N}] [, 機能]]]]]]
```

（水平位置，垂直位置）は，円の中心を示す位置です．

半径は，X 軸上での半径で，ドットの個数で与えます．円の半径を与えるのに，X 軸上と特にことわるのは変だと思われるかも知れませんが，これは，ドット間隔が X 軸方向と Y 軸方向でちがうからで，X 軸のドット数と，同じ長さになる Y 軸方向のドット数の割合は 0.4495 にしなければならないのです．

パレットコードは，描くべき円または円弧の色を指定します．

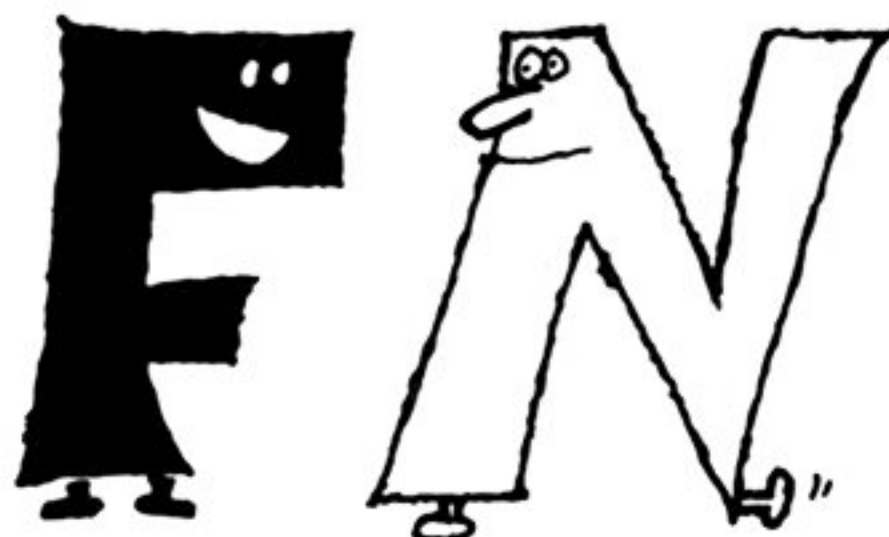
比率は X 軸上のドット数に対する Y 軸上のドット数の比率で，0〜1 の範囲で指定できます．指定がないときは，0.4495 を取って円を描きます．

開始位置は，円の中心位置から X 軸正方向を 0 とし，時計回りに 1 周して再びもとに帰った位置を 1 とするように与えられます．すなわち，0〜1 の間の数が指定できるのです．

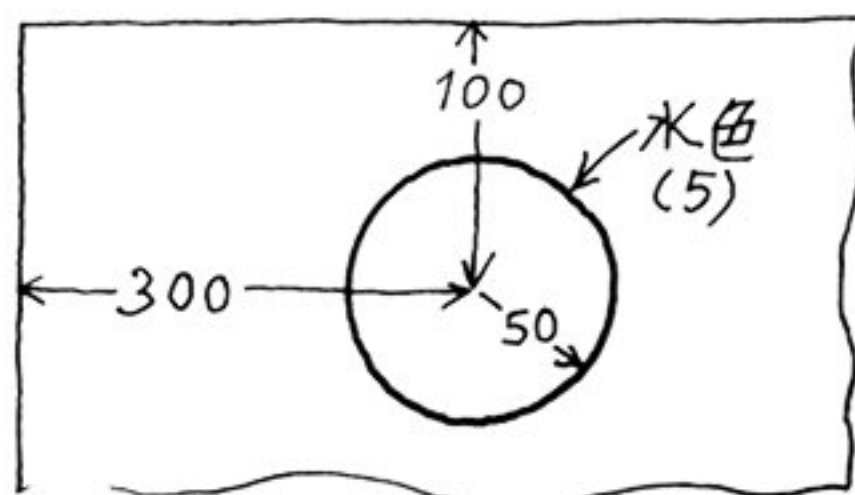
終了位置は描き終わる位置を示し，やはり 0〜1 の範囲で指定します．

開始と終了の位置が等しければ，完全な円が描かれますが，その他の場合は円孤となります．開始，終了位置は省略することができ，そのときは 0 を指定されたものとして X 軸の正方向の点から描き始め，同じ点で描き終わります．

F の指定によって，円の内部を塗りつぶします．N の指定は，塗りつぶしません．省略したときは N と見なされます．

機能には PSET，PRESET，AND，OR，XOR を指定できます．省略したときは PSET が指定されたものと見なされます．

この例は，

CIRCLE @ (300, 100), 50, 5

と入れても結果は同じです．ただし，上例で円の中を塗りつぶす場合は，途中に並んでいるいくつかのコンマは省略できません．

この場合は，縦長の円，つまりだ円になります．

```
CIRCLE@(300,100),50,5,1,,,F
```

では，ここでちょっと複雑なプログラムの例を調べてみましょう．

ここでＳＴＲ＄(Ｒ１)という使い方は，Ｒ１という数値変数を文字変数に変えよ，という命令です．

たとえば，

Ｓ＄＝″RATIO＝″＋ＳＴＲ＄(Ｒ１)

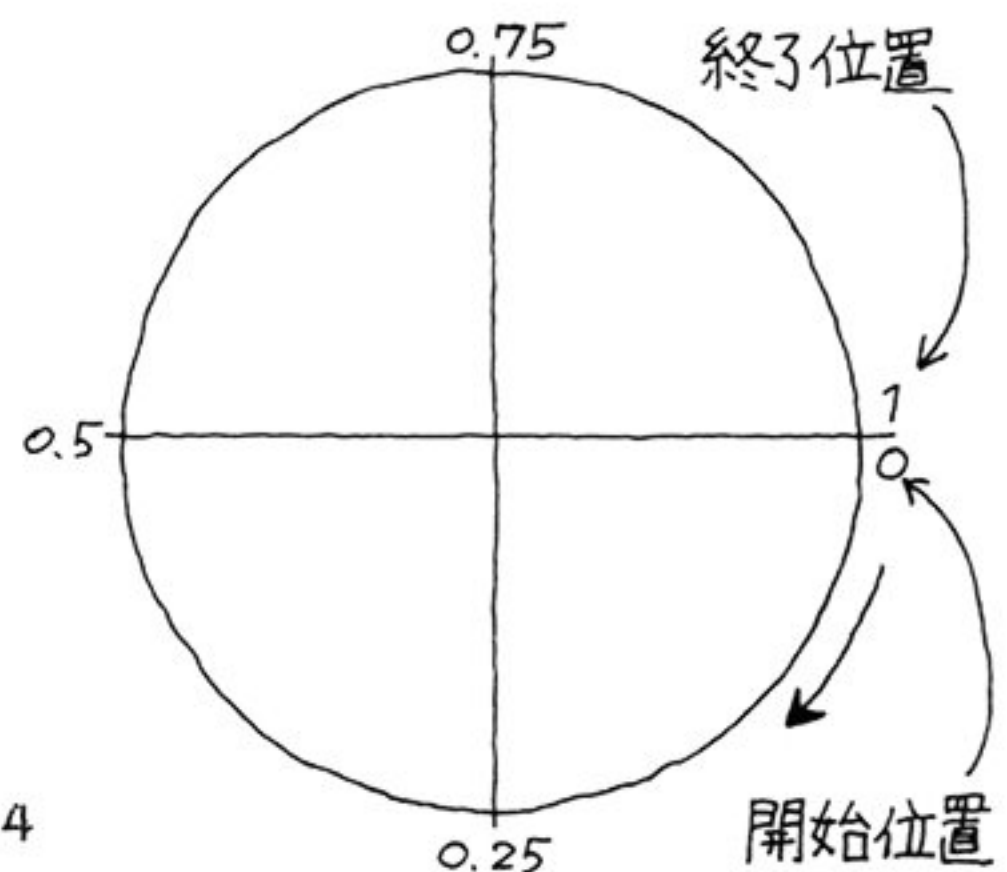
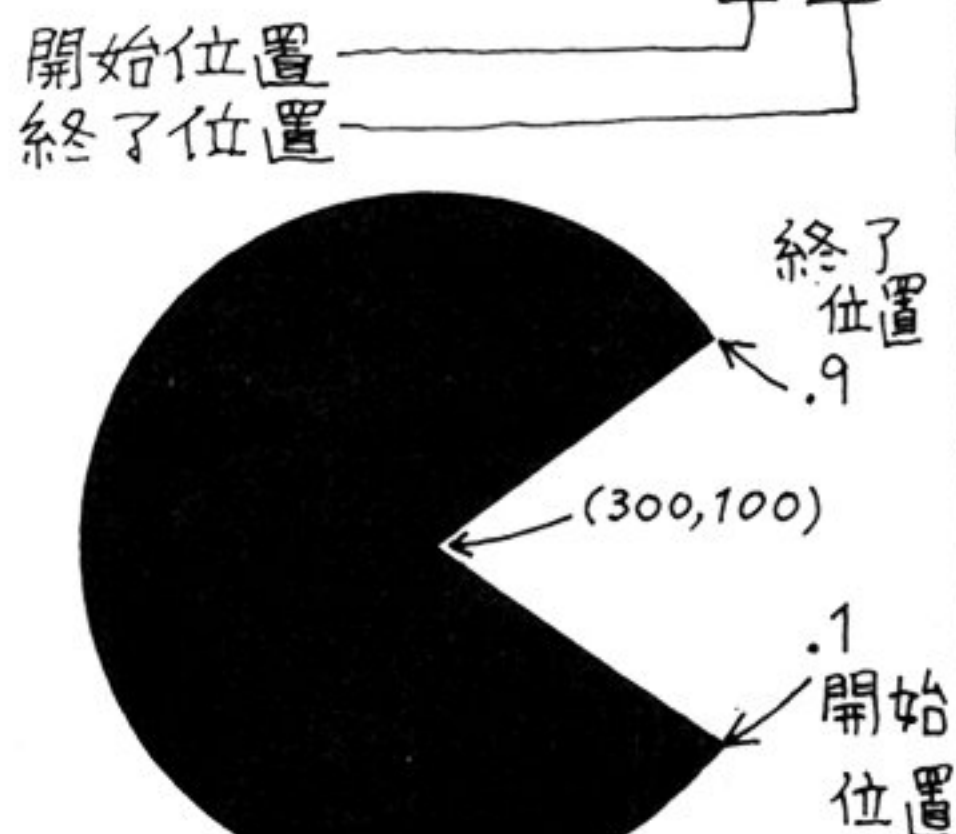
では，Ｓ＄は文字変数です．かりにＲ１＝10としてＳ＄をPRINTすると，

RATIO＝10

として表示してくるはずです．

行番号30はX軸，Y軸の比率Rを0.4495とし，パレットコードを4(緑)，RATIO＝～を表示する位置のうちの，X軸座標（行番号100）を定めています．

40～180がFOR NEXT文．円を描く文は，90と160です．この円の比率はＲ１とＲ２ですが，行番号50，120によって，比率を0.13のI倍ずつ増加させたり，減少させたりしています．同時に行番号70，170で比率を示しています．なお，このあと行番号190，210で比率が1の場合と，0の場合を描いています．

```
10 WIDTH 80,20
20 CONSOLE 0,20,0,0
30 R=0.4495:C=4:X=60
40 FOR I=1 TO 4
50 R1=R+0.13*(I-1)
60 Y=100-100*R1
70 S$="RATIO="+STR$(R1)
80 C1=C-I+1
90 CIRCLE@(300,100),100,C1,R1
100 SYMBOL(X,Y),S$,1,1,C1
110 IF I=1 THEN 180
120 R2=R-0.13*(I-1)
130 Y=100-100*R2
140 S$="RATIO="+STR$(R2)
150 C2=C+I-1
160 CIRCLE@(300,100),100,C2,R2
170 SYMBOL(X,Y),S$,1,1,C2
180 NEXT I
190 CIRCLE@(300,100),100,4,1
200 SYMBOL(60,0),"RATIO=1",1,1,4
210 CIRCLE@(300,100),100,4,0
220 SYMBOL(60,105),"RATIO=0",1,1,4
```

3.10 GCURSOR, PAINT

◆ GCURSOR

カーソルをライトペンと同じように使い，いろいろな指示を与える命令です．このようなカーソルの使い方をグラフィックカーソルといい，画面上には＋印で表示されます．グラフィックカーソルの移動は，キーボード上のカーソル移動キーを使い，座標値のメモリへの読み込みは，RETURN キーが使われます．

GCURSOR　(*x*, *y*), (変数名1, 変数名2) 　　　　　[，(変数名3，変数名4)……] 　　　　　[，パレットコード]

（*x*，*y*）は，最初にグラフィックカーソルを表示する座標です．一度画面に出たら，カーソル移動キーを操作して移動させることができます．

（変数名，変数名）は，グラフィックカーソルの位置の座標値を読み込む変数領域です RETURN キーを押すと読み込まれます．変数名の数でドット座標の個数を指定しますが，全部読み終わると＋印は消えます．なお，読み取る座標の個数は10個が限度です．

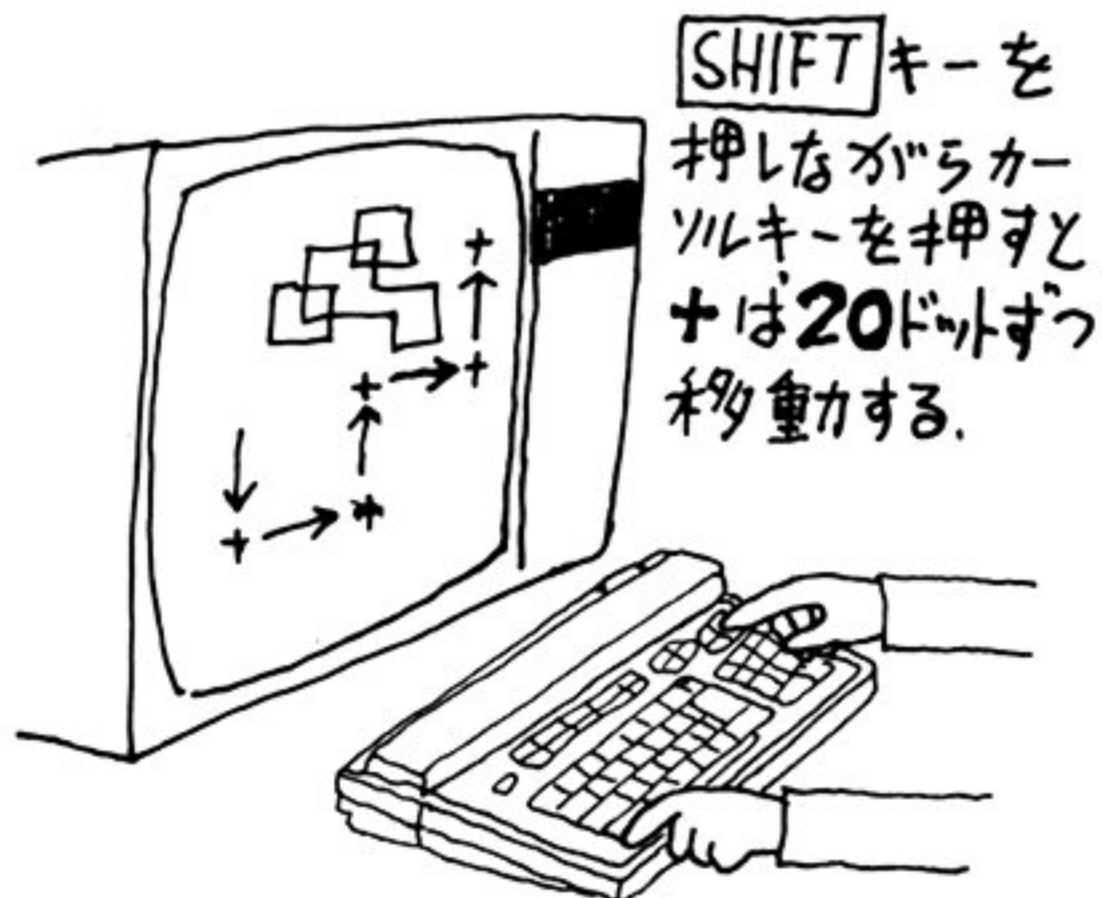
パレットコードはグラフィックカーソルの色を指定します．

◆ PAINT

この命令を実行しますと，指定された境界色で囲まれた範囲を，指定したパレットコードで塗りつぶします．

PAINT　(水平位置，垂直位置) 　　　　[，[パレットコード] 　　　　[，境界色1[，境界色2]……]]

（水平位置，垂直位置）で塗り始めるドットの座標を指定します．その位置を指定すると，その位置が指定した境界色に囲まれた範囲内にあれば，その範囲を指定したパレットコードで塗りつぶします．範囲外にあれば，外側を塗りつぶします．

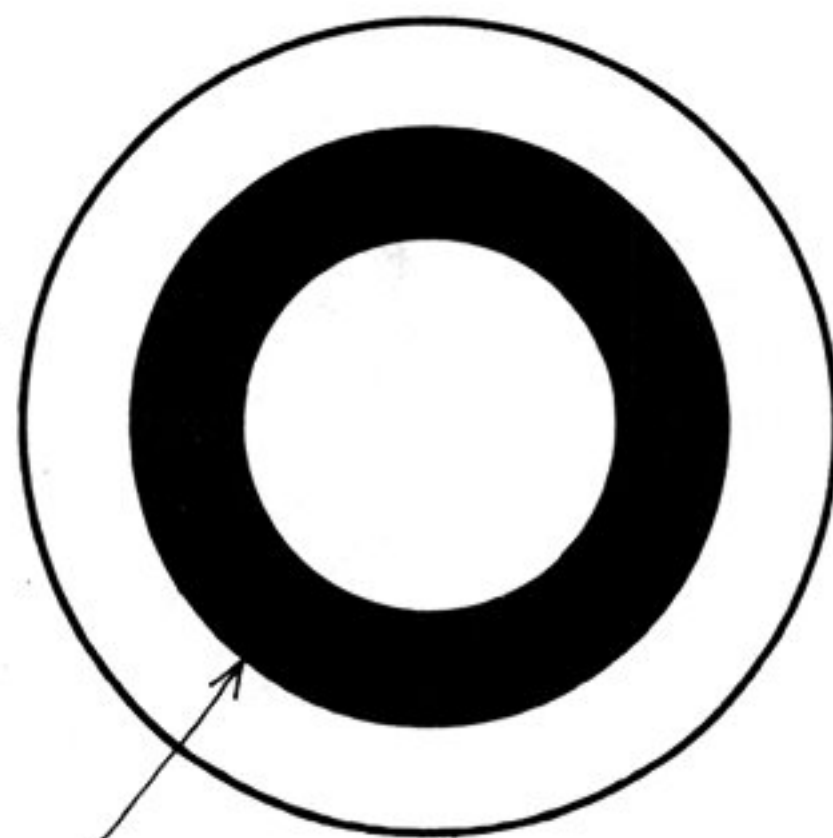
パレットコードを省略した場合，直前に実行した

COLOR文のパレットコードで塗られます．境界色を指定することによって，囲む境界の範囲を指定します．境界色としては複数個を指定できます．

パレットコード自身も境界色の働きをしますから，境界色が指定されていないときは，パレットコードによって境界が指定されます．

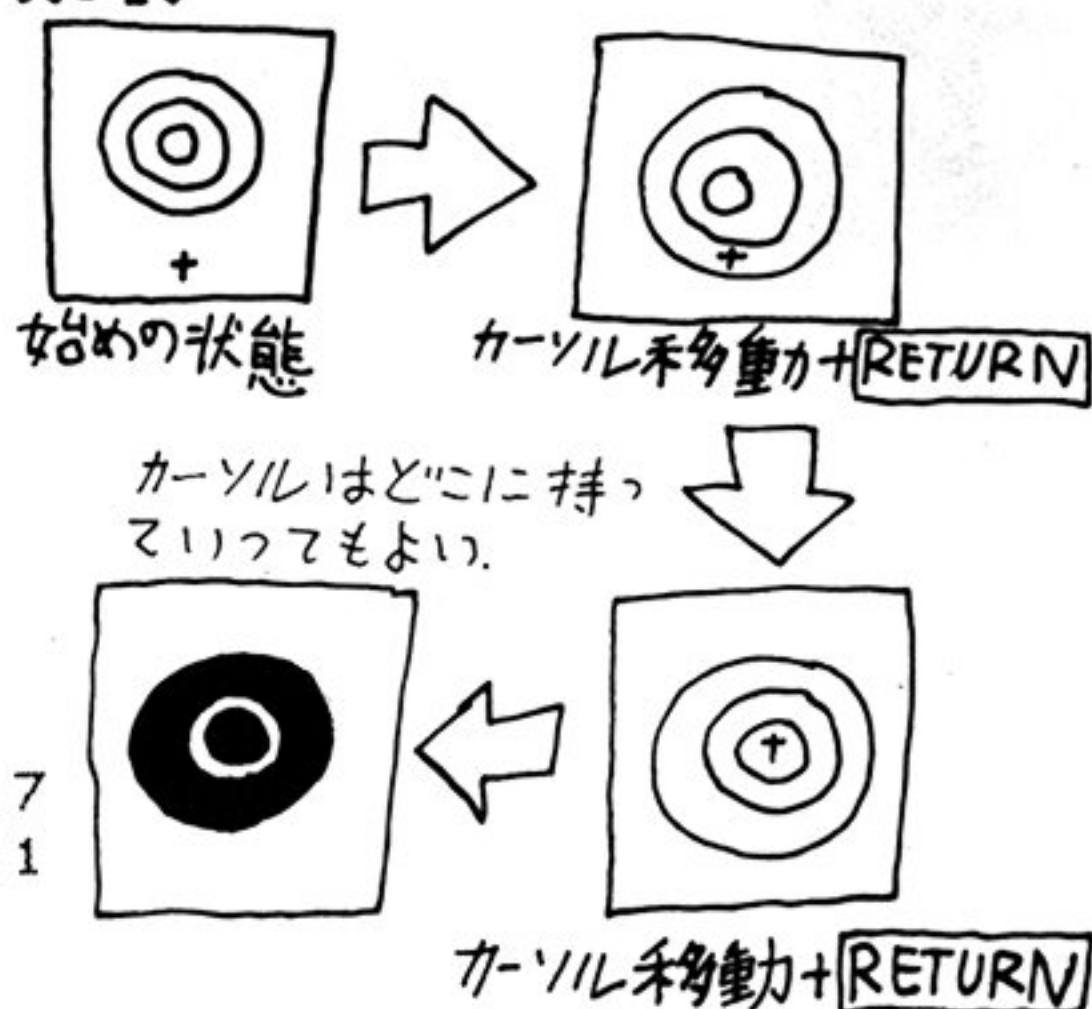
◆ 応用例

簡単なプログラムを使ってこの利用方法を調べてみましょう．

```
10 CLS
20 CIRCLE (300,100),50,2
30 CIRCLE (300,100),70,3
40 CIRCLE (300,100),90,4
50 PAINT (231,100),7,2,3
```

行番号20～40で三つの円からなる同心円を作り，行番号50で円と円との間に色を塗ります．

PAINTは，プログラムの中で色を塗ってしまいましたが，このプログラムは，グラフィックカーソルを動かして指定の位置に色を塗る方法です．もちろんINPUT文を使えば，あなたの好みの色を塗ることも可能です．指定は2回できます．その都度カーソルの色も変わっている点に気をつけてください．

```
10 CLS
20 CIRCLE (300,100),50,2
30 CIRCLE (300,100),70,3
40 CIRCLE (300,100),90,4
50 GCURSOR (300,180),(X1%,Y1%),7
60 GCURSOR (300,180),(X2%,Y2%),1
70 PAINT (X1%,Y1%),6,2,3,4
80 PAINT (X2%,Y2%),5,2,3,4,6
```

```
10 CLS
20 X=320
30 Y=100
40 GCURSOR(320,100),(X,Y),2
50 A$=INKEY$:IF A$="" THEN 50
60 IF A$=CHR$(27) THEN 120
70 IF A$=CHR$(13) THEN 140
80 IF A$=CHR$(28) THEN IF X<639 THEN X=X+1
90 IF A$=CHR$(29) THEN IF X>0 THEN X=X-1
100 IF A$=CHR$(30) THEN IF Y>0 THEN Y=Y-1
110 IF A$=CHR$(31) THEN IF Y<199 THEN Y=Y+1
120 PSET(X,Y,2)
130 GOTO 50
140 END
```

左のプログラムはCRT上にカーソル移動キーを使って自由に線を描き出すプログラムなんじゃー．まだ説明してないステートメントがあるが，そのうち出てくるぞォー．

RUNさせると中央に十印が出る．そこで線を描き出したい位置にカーソル移動キーを使って移動し RETURN キーを押す．そこでカーソル移動キーを押すと線を描き始める．試してごらん．

GCURSORで，十を移動するとき，その前に1～9の数字のキーを押すと，押した数のドットずつ，また0を押すと，10ドットずつ，飛びこえて移動していきます．

3.11 GET @

ここに示すGET @，次のPUT @およびGET文，PUT文の応用は，後述するような配列変数が使われていたりして，一応の予備知識がないと理解に困難な点があります．便宜上ここで説明しますが，場合によりあとまわしにして，ひととおり他の部分を読み終えてからGET @，PUT @に取りかかっても一向にかまいません．

この命令は，画面上のある範囲内に表示されているキャラクタ，あるいはドットを配列変数に読み込むための命令です．PUT @と併用することにより画面を動かしたり，前に求めたグラフを再び出したりすることができ，マイコンゲームなどによく使用する命令です．

この命令文には読み込む形式が四つあります．

読む・変数 ＼ パレットコード	読まない	読　む
キャラクタ	形式 1	形式 2
ドット	形式 3	形式 4

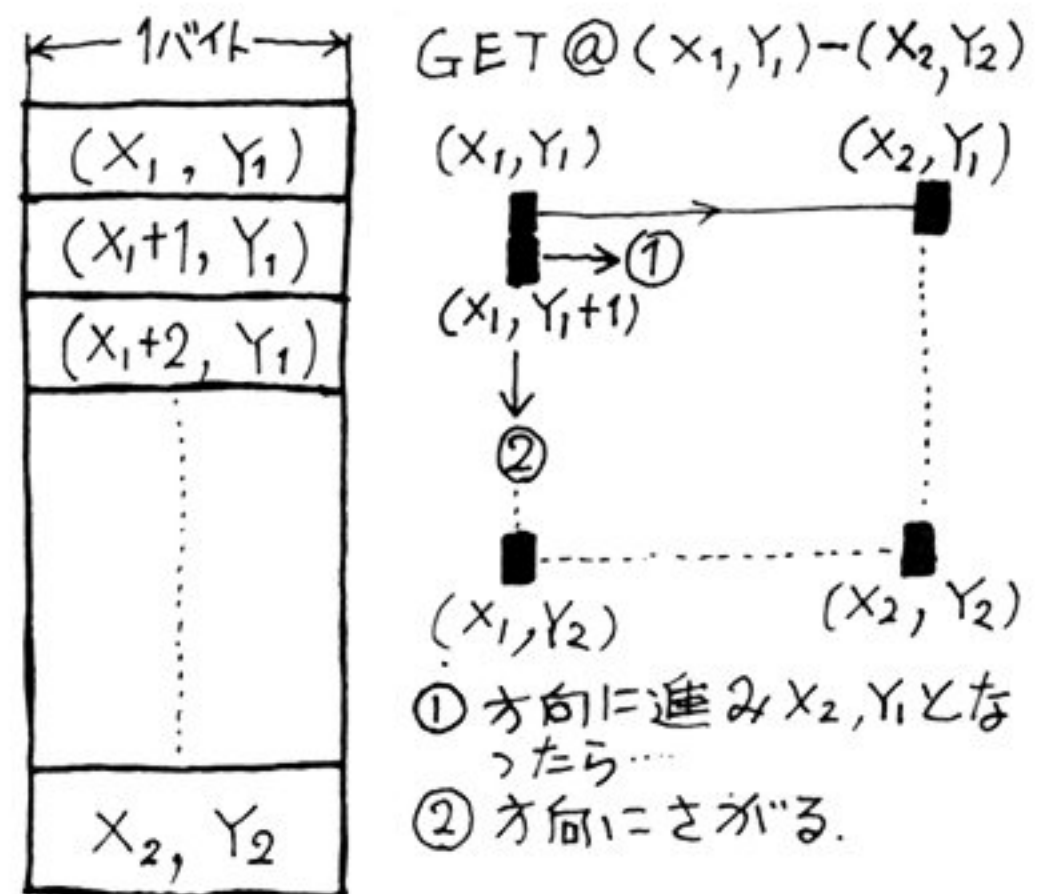
◆ GET @　形式1

画面上に表示しているキャラクタをパレットコードなしで配列に代入します．

GET @ (X_1 , Y_1) － (X_2 , Y_2)，配列名

画面上の任意の領域のキャラクタを配列に読み込みます．(X_1 , Y_1)，(X_2 , Y_2)はキャラクタ単位の座標で読み込む範囲を示します．配列の必要な大きさは，

配列の大きさ＝（読み込む範囲のキャラクタ数＋a－1）／a

aは配列変数一つのバイト数（A%(0)の場合は2バイト）です．読み込む順序は(X_1 , Y_1)から(X_2, Y_1)の方向に進み，次に(X_1, Y_{1+1})となって，再び①の方向に進みます．

ではプログラムを見てみましょう．

```
10 CLS
20 DIMA%(10)
30 LOCATE 15,10
40 PRINT"FUJITSU "
50 GET@(15,10)-(23,10),A%
60 FOR I=0 TO 5
70 PRINT HEX$(A%(I))
80 NEXT I
```

CRT表示例

```
                    FUJITSU
4655
4A49
5453
5520
0
0
```

GET @を使って座標X_1=15，Y_1=10，X_2=22，Y_1=10と指定します．そこには画面上にFUJITSU␣と表示しています．また，A%で整数型を指定しているので，2バイトのエリアを持っています．

そこでA%(0)にFUのキャラクタコード，A(1)にJ Iのコード………A%(3)にU␣のキャラクタコードが代入されます(␣はスペースを示す)．

配列名	読み込まれるキャラクタ	キャラクタコード	16進数	10進数
A%(0)	F, U	46, 55	4655	18005
A%(1)	J , I	4A, 49	4A49	19017
A%(2)	T, S	54, 53	5453	21587
A%(3)	U, ␣	55, 20	5520	21792
A%(4)	なし	なし	0	0
A%(5)	なし	なし	0	0

またGET @ (X_1, Y_1)−(X_2-Y_2) で四角形で囲まれる範囲を指定した場合,

```
10 CLS
20 DIMA%(10)
30 LOCATE 15,10
40 PRINT"ﾊﾟｰｿﾅﾙ   "
50 PRINT TAB(15) "ｺﾝﾋﾟｭｰﾀ "
60 GET@(15,10)-(22,11),A%
70 FOR I=0 TO 8
80 PRINT HEX$(A%(I))
90 NEXT I
```

CRT表示例

```
                ﾊﾟｰｿﾅﾙ
                ｺﾝﾋﾟｭｰﾀ
CADF
BOBF
C5D9
2020
BADD
CBDF
ADBO
C020
0
```

GET @ (15, 10)−(22, 11)と指定してみました.

読み込む範囲が (15, 10) …… (22, 11) まで順にA%に代入されます.

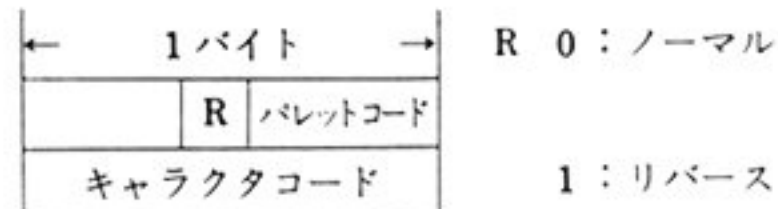
配列名	読み込まれるキャラクタ	16進数
A%(0)	ﾊ, ﾟ	CADF
A%(1)	ｰ, ｿ	BOBF
A%(2)	ﾅ, ﾙ	C5D9
A%(3)	␣, ␣	2020
A%(4)	ｺ, ﾝ	BADD
A%(5)	ﾋ, ﾟ	CBDF
A%(6)	ｭ, ｰ	ADBO
A%(7)	ﾀ, ␣	CO20
A%(8)	なし	0

◆ GET @ A 形式2

画面上の任意の領域のキャラクタをパレットコードと共に読み込みます.

GET @ A (X_1, Y_1)−(X_2, Y_2), 配列名

GET @の形式1と, キャラクタを読み込む範囲の指定などは同じです.

配列の大きさは, 形式1の2倍必要となります. 一つのキャラクタは2バイトのエリアを必要とします.

この命令は表示文字の色まで見るときに使います.

◆ GET @ 形式3

画面上に表示している任意の領域の指定色のドットパターンを配列に読み込みます.

GET @ (X_1, Y_1)−(X_2, Y_2), 配列名 , G[,パレットコード [, パレットコード] ……]

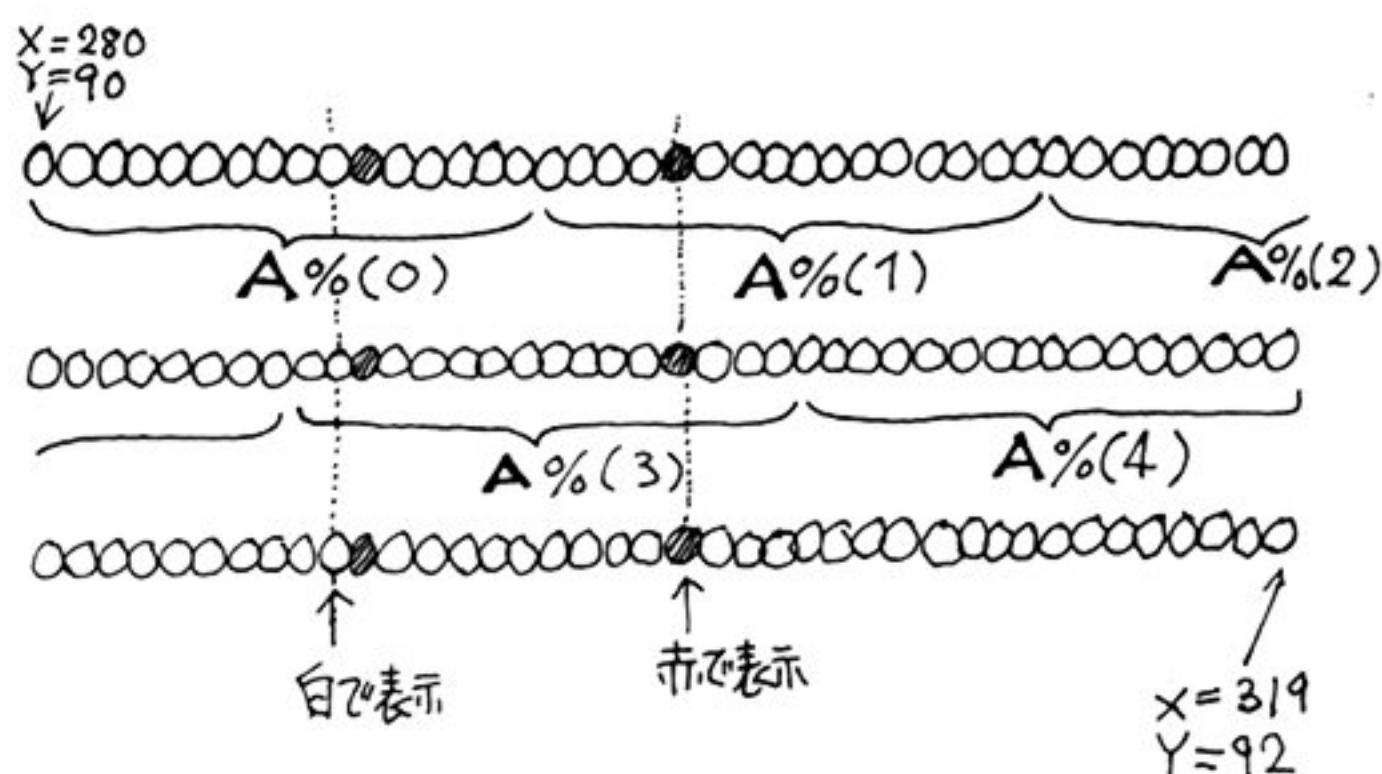
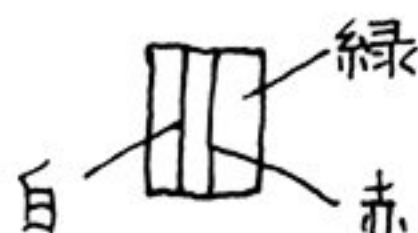
(X_1, Y_1), (X_2, Y_2)はドット単位の座標です.

パレットコードは最大8個まで指定でき, 指定したカラーと, 表示しているドットのカラーが一致していれば, そのビットを1にします.

パレットコードの指定がない場合, 読み込んだデータが背景色以外で表示されていれば, そのビットを1にします.

それではプログラムを実行してみます.

```
10 CLS
20 DIM A%(100)
30 LINE@(280,90)-(319,110)
   ,PSET,4,BF
40 LINE(290,90)-(290,109)
   ,PSET,7
50 LINE(300,90)-(300,109)
   ,PSET,2
60 GET@(280,90)-(319,92)
  ,A%,G,7,2
70 LOCATE 0,0
80 FOR I=0 TO 105
90 PRINT HEX$(A%(I))
100 NEXT I
```

プログラム結果は,

となります.

画面上に緑色の四角形を表示し，その中に白と赤で縦の線が引かれて表示します.

GET @ (280, 90)－(319, 92) でX方向40ドット，Y方向3ドットの領域を指定して，A%の配列に読み込みます. GET文で指定した領域は, 右図のドットで表示しています.

配列名は，A%で整数型を指定していますので, 1配列で2バイト分のエリアがあり，ビットで表現すると16ビット分あります.

このプログラムでは，パレットコードを7（白色), 2（赤色）と指定していたので，白と赤で表示している部分が1となります.

A%(0)はX＝280, Y＝90よりX＝295, Y＝90までのドットを代入します.

A% (0)＝○○○○○○○○○○●○○○○○のドット

A% (0)＝0000000000100000の2進数

A% (0)＝ 0 0 2 0 　　　の16進数

となり，A%(1)はX＝296, Y＝90よりX＝311, Y＝90までのデータを代入します.

◆ GET @ A 形式4

画面上の任意の領域のグラフィックドットを, そのパレットコードと共に配列に読み込みます.

> **GET @ A (X1 , Y1)－(X2 , Y2),**
> **配列名, G**

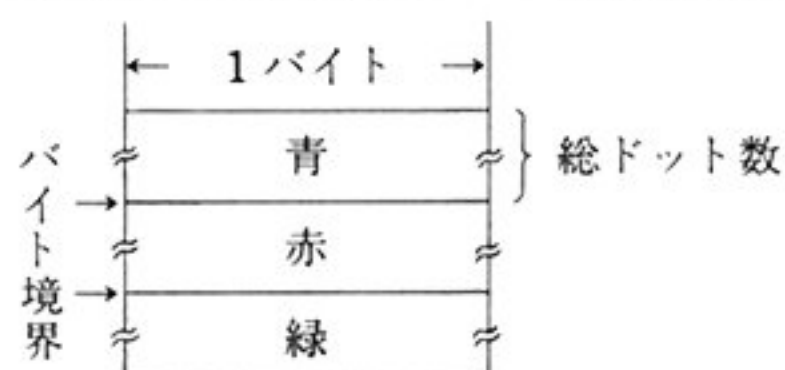
GET @ 形式3は，グラフィックドットを読み込む命令ですが，GET @ A 形式4は, カラーの情報まで読み込む命令です.

(X_1, Y_1), (X_2, Y_2) はドット単位の座標で, 指定方法は形式3と同じです.

読み込むための配列の大きさは,

配列の大きさ＝(((総ドット数＋7)／8)×3＋a－1)／a

aは配列変数一つの使用バイト数

次の形式で配列にデータが格納されます.

各座標のドットがカラーの赤，緑，青のビットの組み合わせによりできているので，その各ビットを青，赤，緑の順に配列へ代入します.

表示色	代入されるドット		
	青	赤	緑
黒	0	0	0
青	1	0	0
赤	0	1	0
紫	1	1	0
緑	0	0	1
水色	1	0	1
黄	0	1	1
白	1	1	1

1ドットを1ビットとして青，赤，緑の配列に代入します.

それではプログラム例を実行してみましょう.

```
10 CLS
20 DIM A%(100)
30 LINE@(280,90)-(319,110)
     ,PSET,4,BF
40 LINE@(290,90)-(290,109)
     ,PSET,7
50 LINE@(300,90)-(300,109)
     ,PSET,2
60 GET@A(280,90)-(319,91)
     ,A%,G
70 LOCATE 0,0
80 FOR I=0 TO 14
90 PRINT HEX$(A%(I))
100 NEXT I
```

このプログラムは，GET＠A 形式4を用いて，画面上の表示しているドットを読み取り，その数値を表示するプログラムです．

GET＠A(280, 90)－(319, 91) でX方向40ドット，Y方向2ドットの全体で80ドットの領域を指定しています．

X＝280
Y＝90
白で表示　赤で表示　その他は緑で表示
A% (0)　A% (1)
A% (2)　A% (3)　A% (4)　X＝319
Y＝91

図のようなドットの領域を指定したことになります．今，青のドットだけ注目しましょう．最初の16ドットを見ると，緑と白で表示しています．緑は緑のドットだけで表示しますが，白は青，赤，緑のドットで表示します．ですから青のドットが関係するのは，白の表示している部分だけなので，最初の16ドットは，A%(0)＝″0000000000100000″ 16進数で0020となります．その次の16ドットは，緑と赤で表示していますので，青のドットはなく，A%(1)＝0となります．

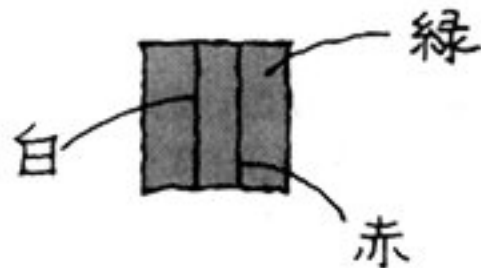
16ドットずつ，次々に代入して，80ドット分代入し終わり，A%(4)＝0となります．次は赤のドットを注目してください．ドットの座標がX＝280，Y＝90より，16ドットがA%(5)＝″0000000000100000″ 16進数で0020となり，その次の16ドットはA%(6)＝″0000100000000000″ 16進数で0800となります．A%(9)＝0の次は緑のドットを注目してください．

ドット座標がX＝280，Y＝90より，16ドットが表示色が緑と白なのでA%(10)＝″1111111111111111″ 16進数でFFFFとなります．次の16ドットは緑と赤なのでA%(11)＝″1111011111111111″ 16進数でF7FFとなります．同様にして次々にデータを読み取り，A%(14)はFFFFとなって画面上のGET＠Aで指定した領域のドットを全て配列に代入します．

A%(0)＝0000000000100000
A%(1)＝0000000000000000
A%(2)＝0000000000000000　40ドット 青のドットデータ
A%(3)＝0010000000000000
A%(4)＝0000000000000000
A%(5)＝0000000000100000
A%(6)＝0000100000000000
A%(7)＝0000000000000000　40ドット 赤のドットデータ
A%(8)＝0010000000001000
A%(9)＝0000000000000000
A%(10)＝1111111111111111
A%(11)＝1111011111111111
A%(12)＝1111111111111111　40ドット 緑のドットデータ
A%(13)＝1111111111110111
A%(14)＝1111111111111111

以上のように配列に，表示データを読み込んできます．

```
20
0
0
2000
0
20
800
0
2008
0
FFFF
F7FF
FFFF
FFF7
FFFF
```

緑
白
赤

プログラムRUNの結果です．ビットパターンと同じ数値になっているか検討してみてください．

3.12 PUT @

この命令は，画面上のある範囲内に配列内の数値をデータとして表示する命令で，GET @ と全く反対の動作をします．

この命令文にも四つの形式があります．

表示する要素＼パレットコード	カラーを指定できる	カラーを指定できない
キャラクタ	形式1	形式2
ド ッ ト	形式3	形式4

◈ PUT @ 形式1

GET @ 文（形式1）で読み込まれたキャラクタを画面の任意の位置に表示します．

> **PUT @ (X1, Y1) − (X2, Y2), 配列色, ［パレットコード］**

GET @ 文で配列に読み込まれたキャラクタを画面上に表示します．

配列にデータを代入して表示する場合は，GET @形式1の書式にしたがって，配列にデータを書き込んでください．

パレットコードは，表示するキャラクタの色を指定します．この指定がない場合は，直前のパレットコードで表示します．

プログラム例

```
10 CLS
20 DIMA%(10)
30 LOCATE 15,10
40 PRINT"FUJITSU"
50 GET@(15,10)-(22,10),A%
60 PUT@(0,0)-(22,10),A%
70 PUT@(10,11)-(10,18),A%,2
80 LOCATE 1,1
```

このプログラムでは，行番号40の文字をGET @文で読み取り，PUT @文で表示させます．行番号60では，文字を横に並べて，また行番号70では縦に並べて表示します．

◈ PUT @A 形式2

GET @ A文（形式2）で読み込まれたキャラクタを，画面上の任意の位置に表示します．

> **PUT @A (X1, Y1) − (X2, Y2), 配列名**

GET @ Aで配列に読み込まれたキャラクタを画面上に表示します．

配列にデータを代入して表示する場合は，GET @ A形式2の書式にしたがって配列にデータを書き込んでください．

PUT @ Aでカラー指定はできません．カラーを変更する場合は，配列中のデータを変更してください．

◈ PUT @ 形式3

GET @（形式3）で読み込んだグラフィックドットを画面に表示する命令です．

> **PUT @ (X1, Y1) − (X2, Y2), 配列名, 機能, ［, パレットコード］**

GET @ 文で読み込んだ配列の中には，ドットのカラー情報はなく，表示するかしないかの情報を持っているだけです．したがって，表示するドットの色を，パレットコードで指定するのです．省略されていれば，直前に実行されたCOLOR文のパレットコードで表示します．

機能としてはPSET, PRESET, AND, OR, XOR, NOTのどれかを指定できます.

例として，次のような場合を考えてみましょう.

	パレットコード	色	ビット
指定	2	赤	010
画面	6	黄	110

それぞれの機能によって，画面に表示される色は右のようになります.

機　　能	ビット パターン	パレット コード	色
PSET	0 1 0	2	赤
PRESET	1 1 0	6	黄
AND	0 1 0	2	赤
OR	1 1 0	6	黄
XOR	1 0 0	4	緑
NOT	1 0 1	5	水　色

プログラム例

```
10 CLS
20 DIM A%(50)
30 LINE@(280,90)-(319,110),PSET,4,BF
40 LINE (290,90)-(290,109),PSET,7
50 LINE (300,90)-(300,109),PSET,2
60 GET@ (280,90)-(319,95),A%,G,7,2
70 PUT@ (10,10)-(49,15),A%,PSET,5
80 PUT@ (100,100)-(138,105),A%,PSET,6
90 PUT@ (104,106)-(144,111),A%,PSET,6
100 LOCATE 1,1
```

GET @ で画面上から白と赤のドットのみ配列に読み込むので，データは || がA%の中にあり，行番号70はそのデータを画面の(10, 10)座標から表示させています.

行番号80，90では，横の表示ドット数を変化させて表示しています.

画面に表示している図形をよく見て検討してください.

◆ PUT@A　形式4

GET @ 文（形式4）で読み込まれたグラフィックドットを任意の画面に表示する命令です.

> **PUT @ A　(X_1, Y_1)－(X_2, Y_2), 配列名，機能**

配列はパレットコードも持っているため，そのまま表示すればよいのですが，機能によって色を変えることができます.

機能としてはAND, OR, XOR, NOT, PSETのいずれかを指定できます.

PSETの指定をしたときは，配列データをそのまま画面に表示します.

NOTを指定したときは，配列の青，赤，緑の全ビットを反転させた結果，すなわち配列に読み込まれている色の補色で表示されます.

AND, OR, XORを指定したときは，現在画面に表示されているドットの青，赤，緑のドットパターンと，配列で示されたドットパターンが画面に表示されます.

3.13 GET文とPUT文の応用例

文字の大きさ
横方向倍率：3
縦方向倍率：2
背景色：黒

図 3.12-1

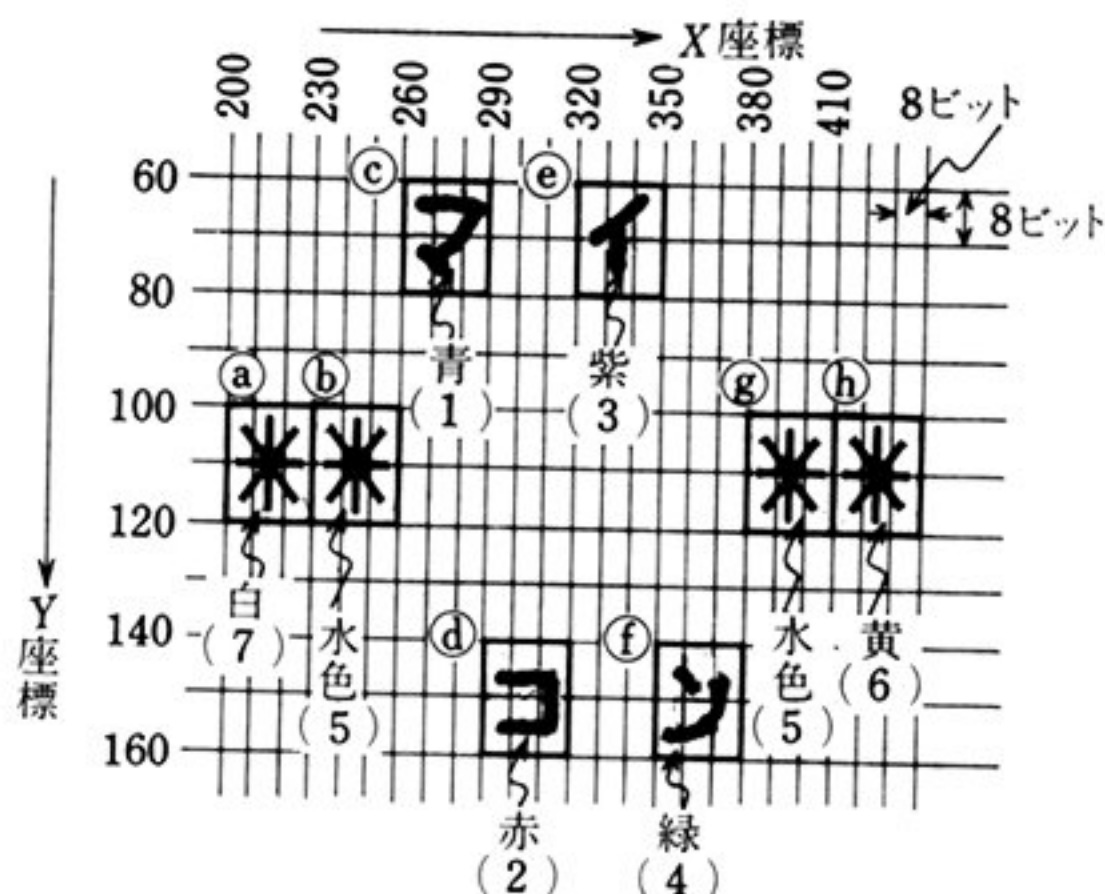
GET と PUT について長々と解説してきましたが，何はともあれ実際に体験してみることが，理解への早道でしょう．

とにかく，実際にプログラムを作って実行してみましょう．

```
10 CLS
20 DIM F%(50),M%(50),H%(50),E%(50)
30 SYMBOL (200,100),"*",3,2,7
40 SYMBOL (230,100),"*",3,2,5
50 SYMBOL (260, 60),"ﾏ",3,2,1
60 SYMBOL (290,140),"ｲ",3,2,2
70 SYMBOL (320, 60),"ｺ",3,2,3
80 SYMBOL (350,140),"ﾝ",3,2,4
90 SYMBOL (380,100),"*",3,2,5
100 SYMBOL (410,100),"*",3,2,6
110 GET@ (260, 60)-(290, 80),F%,G,1
120 GET@ (290,140)-(320,160),M%,G,2
130 GET@ (320, 60)-(350, 80),H%,G,3
140 GET@ (350,140)-(380,160),E%,G,4
150 FOR Y=60 TO 96 STEP 4
160 PUT@ (260,  Y)-(290,Y+20),F%,XOR,1
170 PUT@ (260,Y+4)-(290,Y+24),F%,XOR,1
180 PUT@ (290,200-Y)-(320,220-Y),M%,XOR,2
190 PUT@ (290,196-Y)-(320,216-Y),M%,XOR,2
200 PUT@ (320,      Y)-(350,Y+20),H%,XOR,3
210 PUT@ (320,  Y+4)-(350,Y+24),H%,XOR,3
220 PUT@ (350,  200-Y)-(380,220-Y),E%,XOR,4
230 PUT@ (350,  196-Y)-(380,216-Y),E%,XOR,4
240 NEXT
```

20 は配列変数を定義しています．それぞれの変数には，50 ～ 80 のステートメントで画面にドットパターンで表示した〝**マ**〟，〝**イ**〟，〝**コ**〟，〝**ン**〟の4文字のドット情報が読み込まれます．

30 では@の点のドット座標で，〝*〟を横倍率3，縦倍率2，パレットコード7(白)で表示します．40 ～100も同様のステートメントです．

110は，〝**マ**〟のドットパターンを形式3のGET@ステートメントで整数型配列変数F%に読み込みます．〝**マ**〟は50でわかるように，パレットコード1（青）で表示しているので，そのままのパレットコードで読んでいるのです．

120 ～ 140も同様のステートメントです．

160 ～ 240はどのステートメントも同型式で，形式3の PUT @ を使っています．そして，2行ずつF%，M%，H%，E%の配列変数を PUT @ しています．機能は全てXORとなっていますね．

160 の PUT @ ステートメントでは何をしているのでしょう．この直前に画面に表示されているパレットコードと，このステートメントで指定しているパレットコードについて XOR の論理演算をしていますから，ビットパターンは〝000〟となり，背景色と同じになります．すなわち，画面から消えるわけです．

そして，170 の PUT @ ステートメントでY座

標を4ドットずらして表示しています．

180と190，200と210，220と230の組も全く同じ論理のステートメントです．そして，全部の文字が一直線に並んで（Y＝96）終わりになります．

ところで，20 DIM文の配列数の決め方をまだ解説していませんでしたね．では，前ページ図の〝マ〟に注目してください．そして，3.10の形式3のGET @文で説明した〝配列の大きさ〟の式をみてください．〝マ〟の箱の中の総ドット数は，(290－260＋1)×(80－60＋1)＝651になります．配列変数F%は整数として定義していますから，1要素の大きさaは2バイトです．そのため，

配列の大きさ＝((651＋7)/8＋2－1)/2
　　　　　　＝41　　（小数部分切り捨て）

となります．配列の大きさは，少し大きき目に取ってもかまわないため，〝50〟としたのです．

マ　コ

＊＊　　　＊＊

イ　ン

＊＊マ　コ＊＊
イ　ン

＊＊マイコン＊＊

4 いろいろなステートメント

この章では，BASICを利用する上で，ぜひ知っておかなくてはならないいろいろなステートメントを中心に説明を進めていきたいと思います．

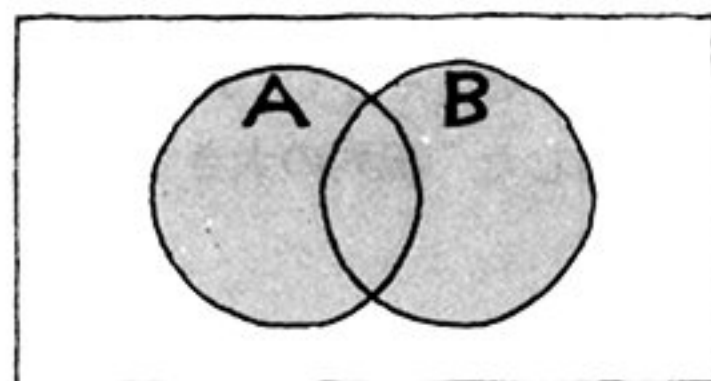
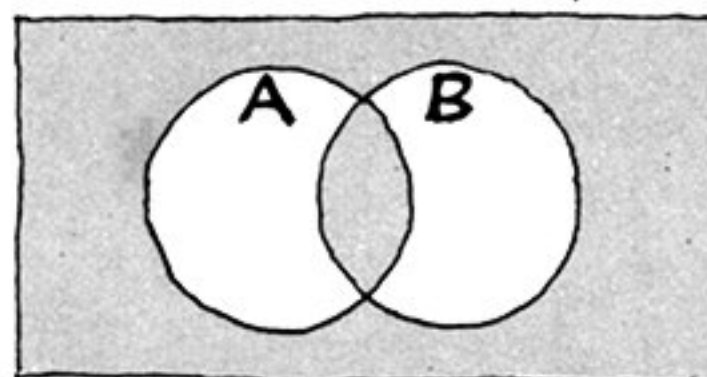
コンピュータは，判断によってイエスかノーの結果を出す仕事が少なくありません．関係式もそうでしたね．ここでは，もう一つの判断をする論理式というものからまず説明を進め，一般ステートメントに移っていくことにします．

4.1 論理式

すでに，もう何回か出てきましたが，ビット操作やブール演算を実行したり，いくつかの関係式を調べたりするときに，論理式が用いられます．ここでは，論理演算子として使われるNOT，AND，OR，XOR，IMP，EQVなどがどんなものかを具体的に調べながら説明を進めます．

ブール演算というのは，19世紀英国のジョージ・ブールという数学者が作り出した，論理的な関係を表現するための演算の方法です．コンピュータは，全て2進数で動いています．そして，論理式を演算するときは，16ビットで動いています（10進数に直すと−32768〜32767までの数）．

そのため，それ以上の数をインプットすると，エラーになることをおぼえておいてください．またマイナスの数は，補数表示という方法がとられています．たとえば，10進数の−1は1111111111111111，−2は1111111111111110，−3は1111111111111101です．なお，16進数の第1桁目は1なら負，0なら正の符号を示しています．

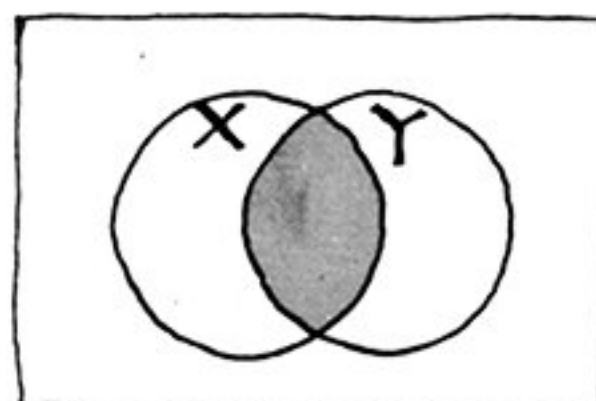
論理式では，各ビットごとにそれぞれ演算をし，その結果を出してきます．それぞれ各桁ごとに演算をし，桁上がりなどしない計算の方法です．な

Aが1であっても，Bが1であっても，結果は1になる，という意味．

AとBが1である場合か，どちらも1ではない場合だけ結果が1になる．

このような図を，ベン図という．

お，インプットする数値が整数でない場合には，小数点以下を四捨五入して，整数に直してから演算をします．

むずかしい話は抜きにして，実際の例でその働きを見てください．

◆ X AND Y（論理積）

ANDというのはXであり，かつYである，という意味です．

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

このような表を真理値表といいます

 の部分がX AND Y

2 AND 3を調べてみましょう.
2進数に直すと,

2は	0000000000000010
3は	0000000000000011
2 AND 3は	0000000000000010

上を見ると，各桁で両方共1になっているところは，下から2桁目だけですから0010つまり結果は2になります.

PRINT 2 AND 3 RETURN
2

いくつかの例を調べてみましょう.

PRINT 3 AND 4 RETURN
0

10進数の3は2進数では11， 4は100ですから，ANDの結果は0になってしまいます.

PRINT －1 AND 4
4

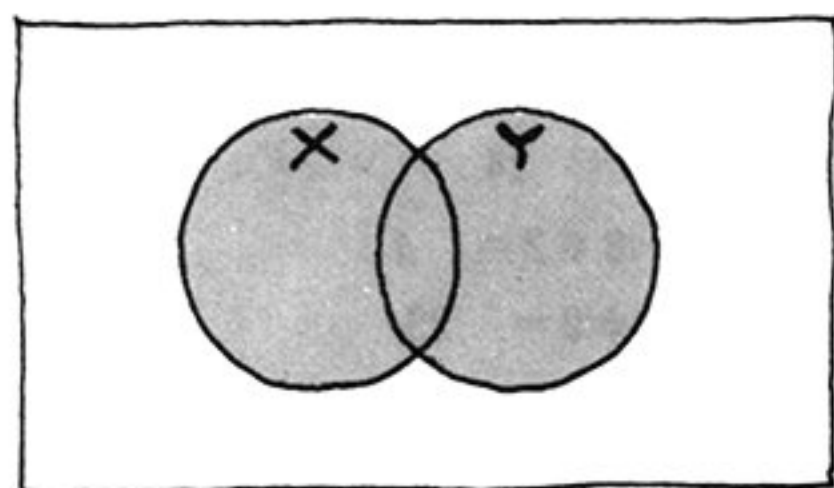
－1は	1…1111
4は	0…0100
－1 AND 4は	0…0100

PRINT －1 AND －2
－2

－1は	1…1111
－2は	1…1110
－1 AND －2は	1…1110

◆ X OR Y（論理和）

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

XまたはYという意味です.ですから，XかYのどちらかが1であれば結果は1になる，という意味です.

PRINT 1 OR 2
3

1は	0…0001
2は	0…0010
1 OR 2は	0…0011

PRINT －1 OR －4
－1

－1は	1…1111
－4は	1…1100
－1 OR －4は	1…1111

PRINT 4 OR 11
15

4は	0…0100
11は	0…1011
4 OR 11は	0…1111

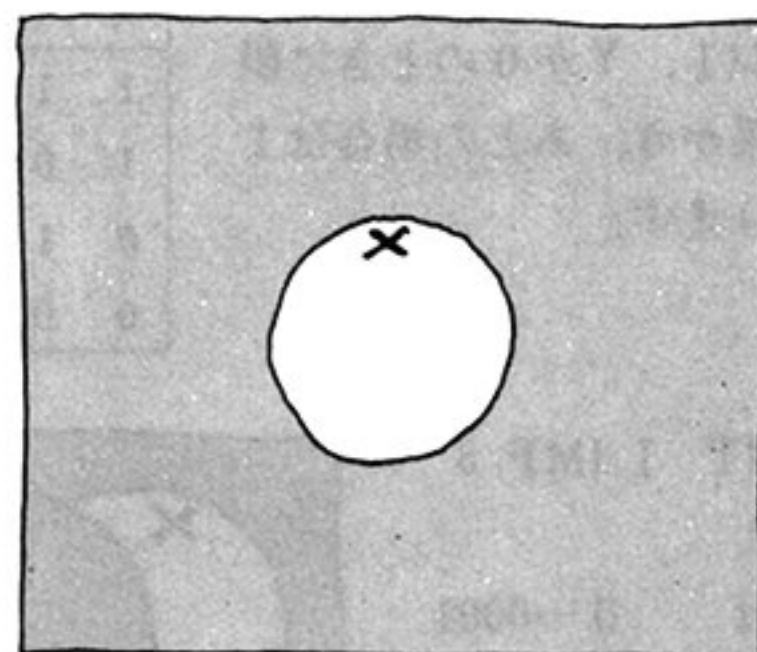
◆ NOT X（否定）

X	NOT X
1	0
0	1

Xではないという意味です.

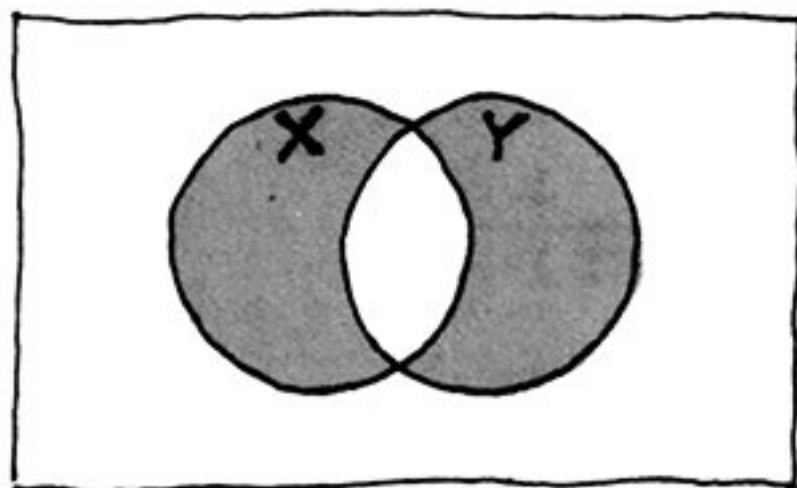
PRINT NOT 1
－2

1は2進数で…0001，その否定だと…1110ですから，－2になるわけです.
では逆はどうでしょう.

PRINT NOT －2
1

◆ X XOR Y（排他的論理和）

XかYのいずれかが1のときだけ，結果が1になるのがXORです．

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

```
PRINT  1 XOR 2
 3
  1は      0…0001
  2は      0…0010
1 XOR 2は0…0011
```

```
PRINT  4 XOR －3
 －7
  4は      0…0100
－3は      1…1101
4 XOR －3は1…1001
```

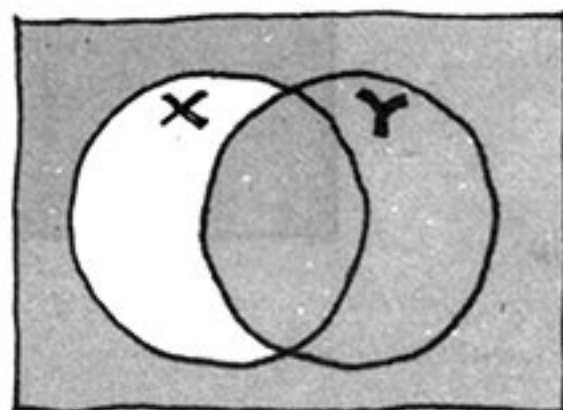
```
PRINT  －5 XOR －3
 6
－5は      1…1011
－3は      1…1101
－5 XOR －3は0…0110
```

◆ X IMP Y（包含）

Xが1，Yが0のときに限り結果が0，あとの場合は1になります．

X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1

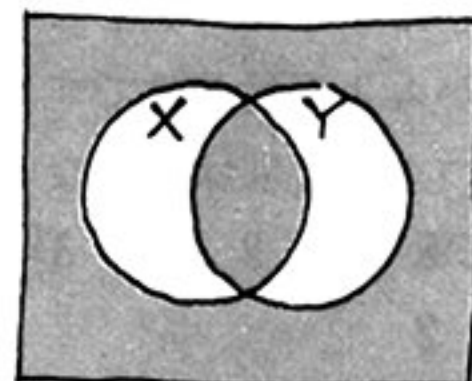
```
PRINT  1 IMP 3
 －1
  1は      0…0001
  3は      0…0011
1 IMP 3は1…1111
```

```
PRINT  3 IMP 1
 －3
  3は      0…0011
  1は      0…0001
1 IMP 3は1…1101
```

```
PRINT  3 IMP －1
 －1
  3は      0…0011
－1は      1…1111
3 IMP －1は1…1111
```

◆ X EQV Y（同値）

X，Yが同じ値であれば1，同じ値でなければ0になります．

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

```
PRINT  5 EQV －3
 7
  5は      0…0101
－3は      1…1101
5 EQV －3は0…0111
```

◆ 優先順位

論理演算は，他の演算子をまぜて使うことも可能です．その場合，例によって演算子の優先順位が問題になります．ここでは，すでに述べてきた演算子を含め，優先順位を表にして示します．

優先順位	演　算　子
1	^
2	－（マイナス符号）
3	*，／
4	¥
5	MOD
6	＋，－
7	関係演算子
8	NOT
9	AND
10	OR
11	XOR
12	EQV
13	IMP

かっこで囲まれた式は，この優先順位に関係なく先に実行されます．

では，いくつかの例で実際に試してみましょう．

```
(5EQV－3)＊4＝  28
5EQV－3＊4＝  14
6＋2/3AND4＋8＊2＝  4
6＋2/3OR4＋8＊2＝  23
6＋2/3XOR4＋8＊2＝  19
6＋2/3IMP4＋8＊2＝－4
6＋2/3EQV4＋8＊2＝－20
1OR2AND3XOR4IMP5EQV6＝－4
NOT(1AND2XOR(3IMP4))＝  3
```

4.2 DEF FN

ちょっとGOSUBに似たところのあるステートメントです．これによって，常時使用する関数の式を，あたかもサブルーチンのようにして使うことができます．GOSUBは次項で説明します．

DEF FN名前[(パラメータリスト)]=式

FNはfunction（関数），DEFはdefine（定義する）の略です．つまり，関数を定義してやるという意味です．

ごく簡単な例で調べてみましょう．

```
10 DEF FNA(X,Y)=X+Y
20 I=2:J=3
30 Z=FNA(I,J)
40 PRINT Z
50 T=10:U=5
60 X=FNA(T,U)
70 PRINT X

RUN
 5
 15
```

行番号10でFNAを定義しています．この関数が呼び出されるのは，行番号30と，60です．ちょっと気になるのは，行番号30の変数はI，J，そして行番号10では，そこがX，Yとなっている点です．実はX，Y（パラメータリスト）のことを仮引数（かりひきすう）といい，I，Jと変数の型（文字変数か，数値変数か）数さえ同じなら，いっこうにかまわないのです．

前に変数とは，文字や数値を入れておく箱と書きました．行番号10は，いわば加工工場であり，外部から運ばれてきた数値は，加工工場では工場専用の箱の中に入れかえられます．この工場専用の箱が仮引数というわけです．

引数を使うと，メインルーチンでいろいろな変数が使い分けられていても，仮引数に置きかえられるため，いっこうにかまわないという利点が生じます．なお，仮引数の中に入るデータの方を実引数（じつひきすう）といいます．また仮引数は，同じ名前の変数

がメインルーチンに使われていても，とにかくその加工工場専用のため，混乱したりすることはありません．

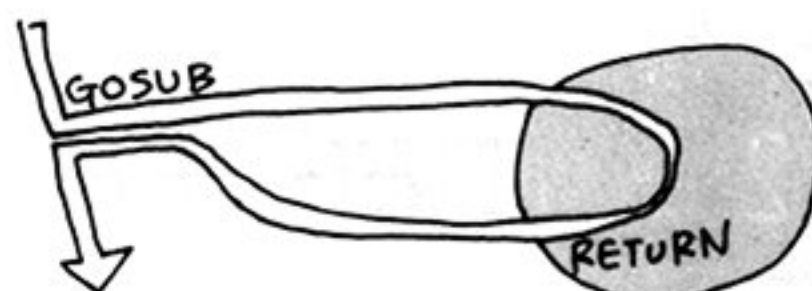
DEF FNは，サブルーチンの場合のように，メインルーチンのあとの方に出てくると使えず，メインルーチンに入る前に，この定義づけをコンピュータに理解させておかねばなりません．また，直接モードでは使うことができません．もちろん，文字変数の場合にも使用できます．

```
10 DEF FNA$(X$,Y$)
      =X$+" "+Y$
20 A$="ﾊﾟｰｿﾅﾙ"
30 B$="ｺﾝﾋﾟｭｰﾀ"
40 PRINT FNA$(A$,B$)

RUN
ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ
```

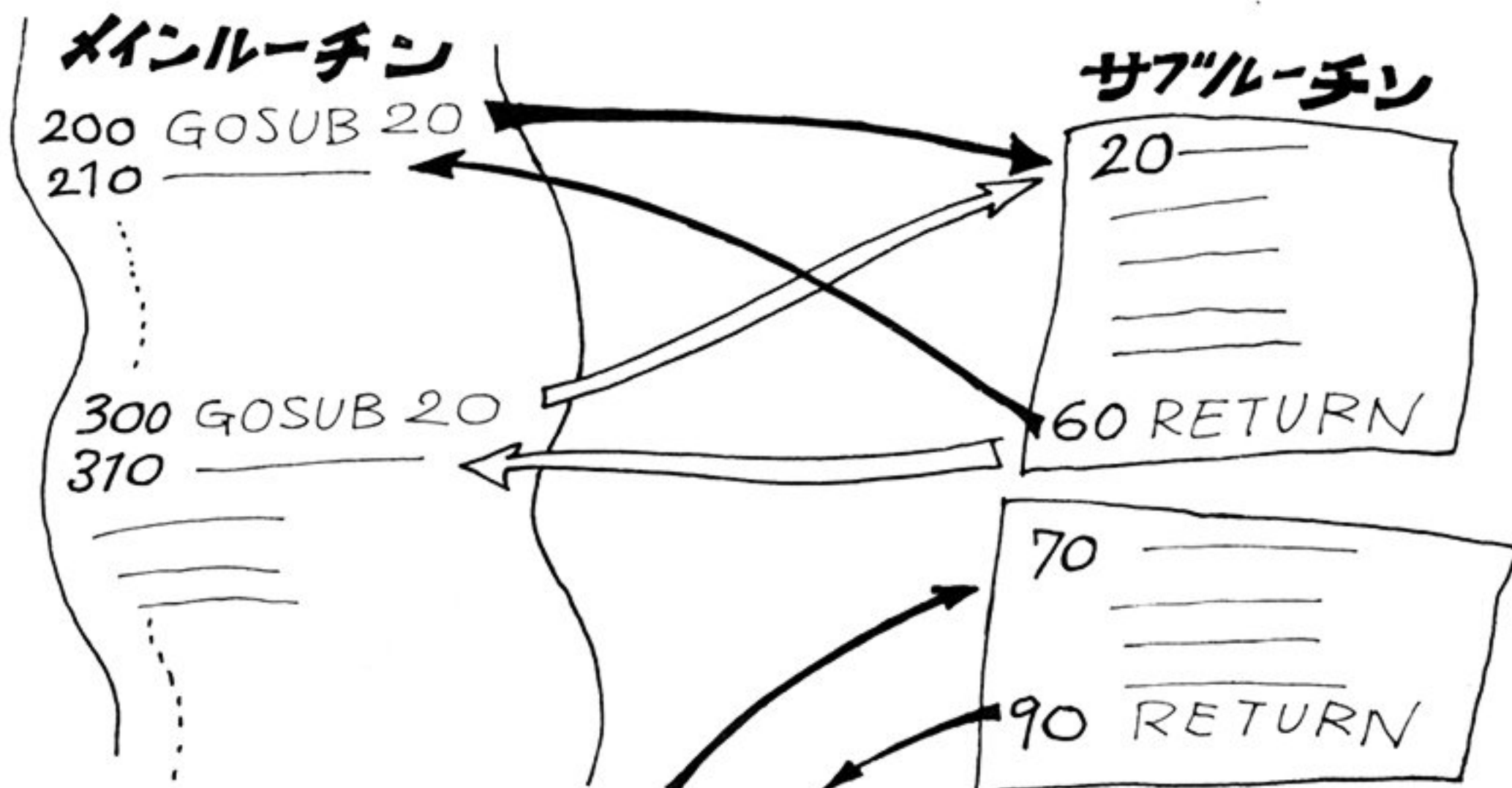
4.3 GOSUB

もっともコンピュータらしい命令の一つです．GOSUB というのはサブルーチン (subroutine) に進め！ という意味です．サブは従，ルーチンは一定の手順，プログラムのことですね．メイン(主)ルーチンに対する言葉と考えてもよいでしょう．プログラムを作るとき，同じ命令ですむ部分が何回も出てくるときに使います．

たとえば，メインルーチンを行番号100～にとっておき，サブルーチンを行番号20～60だとします．

（1） 行番号100から番号順にプログラムが進む．
（2） 行番号200にGOSUB 20の命令がある．
（3） 行番号20にジャンプし，再び番号順にプログラムが進む．
（4） 行番号60でRETURNという命令に出会う．
（5） 再びジャンプし，初めの行番号200の次の行番号の命令に帰ってくる．
（6） 行番号300で再びGOSUB 20の命令に出会う．
（7） 前述（3），（4）が繰り返される．
（8） 再びジャンプし，初めの行番号300の次の行番号に帰ってくる．

この例でわかるように，同じ命令でよい部分が何回も出てくるとき便利なのが，GOSUB～RETURNです．もちろん，サブルーチンはいくつあってもかまいません．たとえばGOSUB 70で，RETURNが行番号90にあってもかまわないわけです．またサブルーチンは，行番号のどの位置にあってもかまいませんが，一般にはサブルーチンがいくつもある場合，ひとかたまりにしてまとめておいた方が，あとで見直すとき便利です．では，簡単な例を作って調べてみましょう．

```
10 GOTO 100
20 FOR I=1 TO X
30 PRINT Y;
40 NEXT I
50 PRINT
60 RETURN
100 X=5
110 Y=1
200 GOSUB 20
210 X=7
220 Y=3
300 GOSUB 20
310 PRINT"オワリ"
RUN
 1  1  1  1  1
 3  3  3  3  3  3  3
オワリ
```

行番号の若い所にサブルーチンを置いた方が処理速度が少し速くなるぞ.

ごく簡単なプログラムです．RUNにより，初めに1を5個，続いて3を7個サブルーチンによって表示してくるはずです．同じサブルーチンでも，変数の違いによって異なる仕事をしてくれることがわかります．別のプログラムを追加してみましょう．

```
45 IF X<=5 THEN GOSUB 70
70 PRINT"アイウエオ"
80 RETURN
RUN
 1   1   1   1   1  アイウエオ
 3   3   3   3   3   3   3
オワリ
```

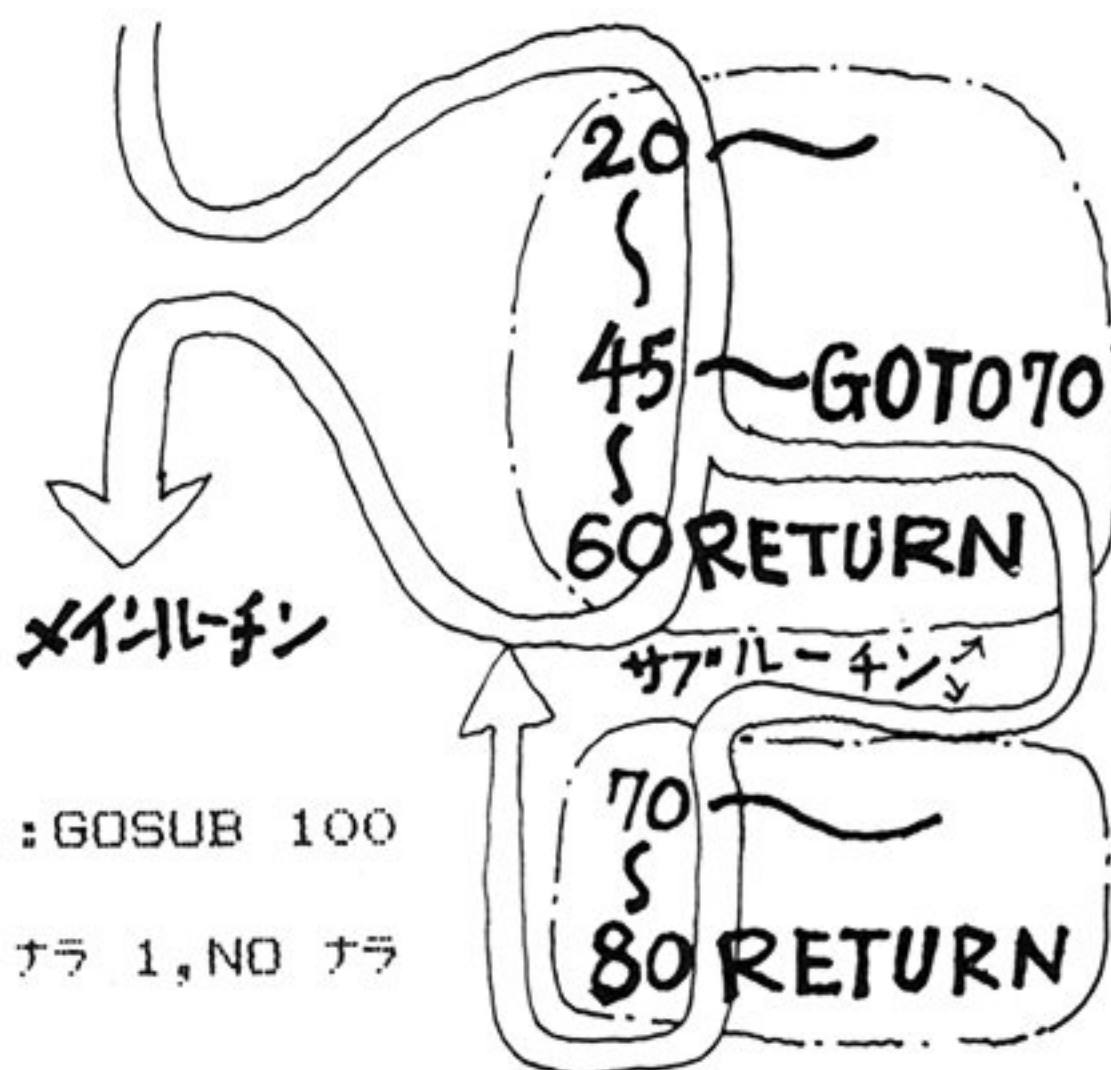
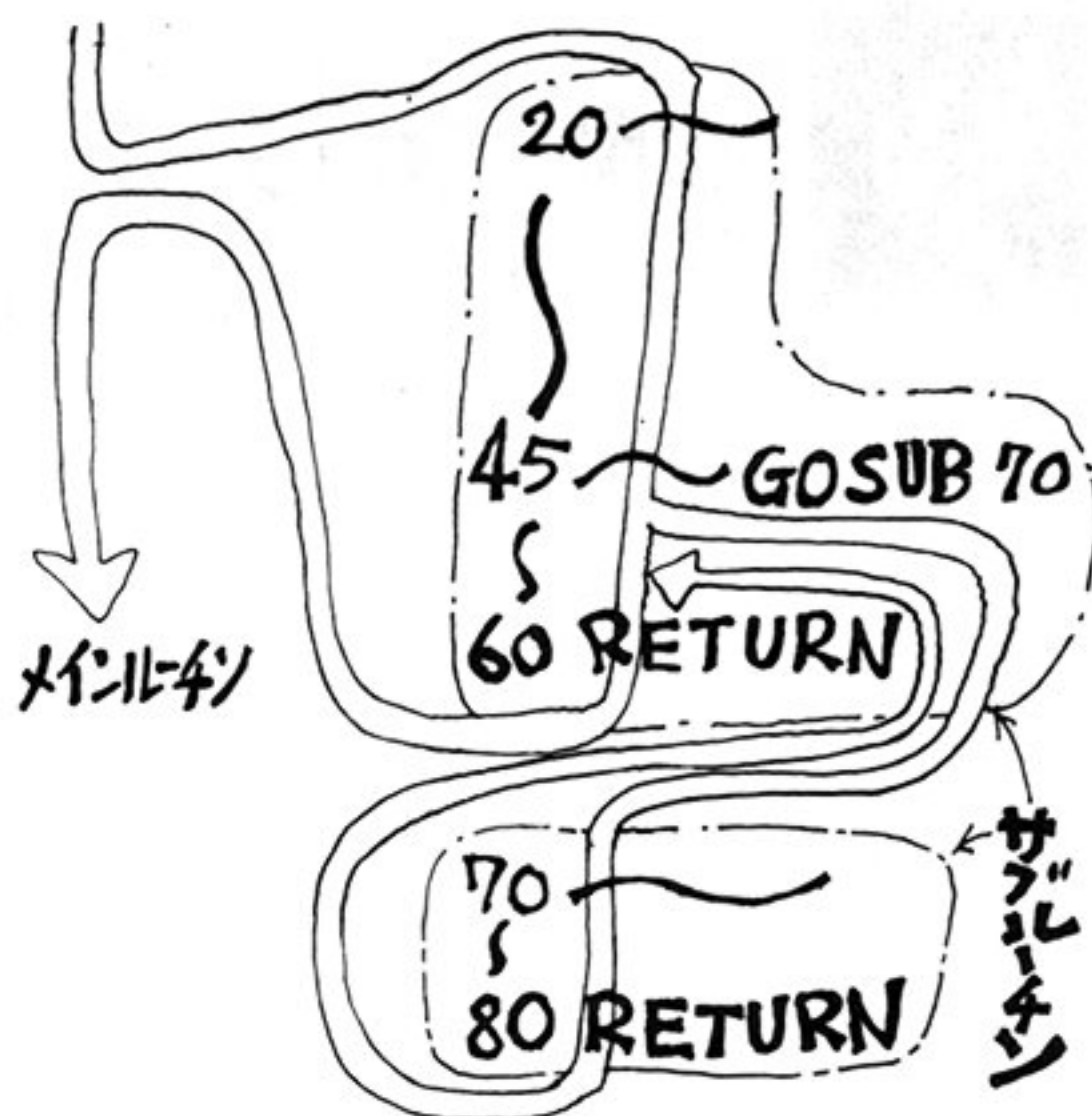
サブルーチンの中にまたサブルーチンがあっても，ちゃんと実行してくれます．行番号45の中のGOSUB 70の部分を，GOTO 70としても実行します．この場合も，80 RETURNは必要です．

◆ 円の面積を計算するプログラムです．

```
10 CLS:INPUT"ハンケイ ハ ";R
20 PRINT:PRINT"エン ノ メンセキ ハ ";:GOSUB 100
30 PRINT" デス":PRINT
40 PRINT"モット ケイサン シマスカ ? YES ナラ 1,NO ナラ
0 ヲ イレテ クダサイ ";
50 INPUT I
60 IF I=1 THEN GOTO 10
70 IF I=0 THEN GOTO 90
80 GOTO 50
90 PRINT:PRINT"オワリ":END
100 X=3.14159*R^2
110 PRINT X;
120 RETURN
```

行番号100～120がサブルーチンです．行番号20からジャンプしてくる部分です．

行番号20, 30, 90にある，独立したPRINT命令は，これによって1行分行を飛ばし，CRT画面上で読みやすくするためのものです．

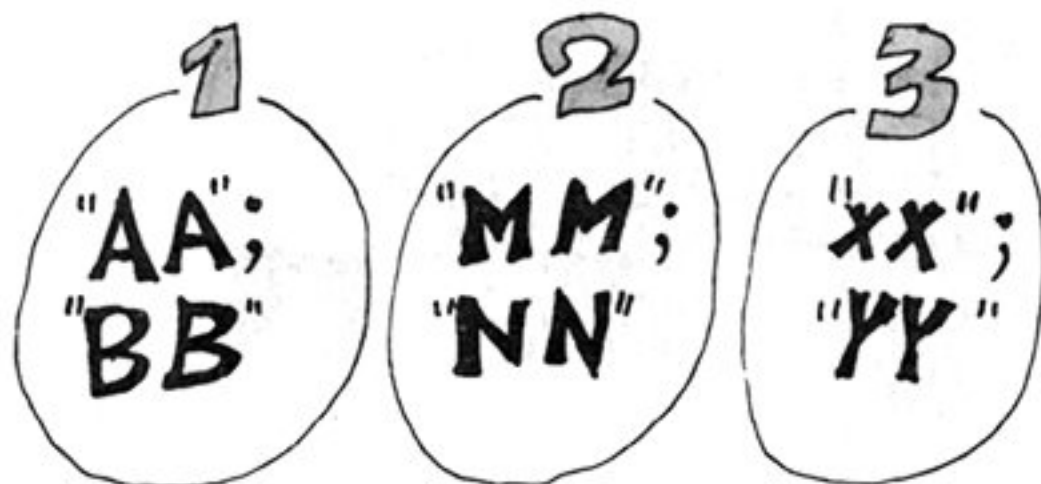
なお，90の終わりにENDが入っていますが，もしこれがないと，プログラム上引き続きサブルーチンのところまで実行してしまいます．

4.4 ON～GOSUB

ON式　GOSUB　行番号[, 行番号]…

ON～GOTO とほとんど同じ働きをします．式に 1 が入ったら，GOSUB のあとに並ぶ行番号の先頭のサブルーチンへ，2 なら 2 番目，3 なら 3 番目です．例を調べてみましょう．

```
10 INPUT X
20 ON X GOSUB 100,200,300
30 GOTO 10
100 PRINT"AA";
110 PRINT"BB"
120 RETURN
200 PRINT"MM";
210 PRINT"NN"
220 RETURN
300 PRINT"XX";
310 PRINT"YY"
320 RETURN
```

? に対して 1 と入れると AABB，2 を入れると MMNN と表示してきます．式が0または行番号の個数よりも大きいときは，次の行に進みます．なお式の値が整数でない場合は，小数部分の第 1 位の桁が四捨五入されて，整数に変換されます．

```
RUN
? 1
AABB
? 2
MMNN
? 3
XXYY
? 4
? 1
AABB
```

4.5 READ～DATA

変数に，プログラム内に置いてあるデータを読み込んでやるとき使うのが，このステートメントです．

READ　変数名[, 変数名]……
DATA　定数[, 定数]……

変数名や，定数がいくつも並んでいるときは，コンマで区切って並べます．定数は文字定数であっても，数値定数であってもかまいませんが，読み込まれる変数の型と一致していなくてはならないことはいうまでもありません．文字定数でスペースや ,（コンマ）を含む定数の場合は，〃〃でくくってやる必要がありますが，その他の場合は必要ありません．

では実例を調べてみましょう．

```
10 DEF FN A(X,Y)=X*Y
20 READ A1,A2,A3,A4
30 Z=FN A(A1,A2):GOSUB 100
40 Z=FN A(A2,A3):GOSUB 100
50 Z=FN A(A3,A4):GOSUB 100
60 DATA 2,3,4,5
70 END
100 IF Z<10 THEN PRINT"ABC"
      ELSE PRINT"XYZ"
110 RETURN
RUN
ABC
XYZ
XYZ
```

行番号60がデータですね．ここに並んでいるデータの数は4個です．これ以上並んでいてもいっこうにかまいませんが，3個のように少ないとOut of Data という表示が出てエラーになります．

たとえば，行番号70をGOTO 20に直してやると，Out of Data になってしまいます．すでに4個のデータを読み込んでしまったからです．そこで，データの数をふやしてやると，データがなくなるまでは動いてくれます．そしてデータがなくなると，やはりOut of Dataになります．

これではおもしろくない，といった場合に使われるのがRESTOREです．RESTOREはもとどおりに復元するという意味です．そこで行番号70を，

70 RESTORE：GOTO 20

としてやれば，このプログラムはいつまでも実行されます．

また，データの最後に実際には使われない値，たとえば−999999などを入れておき，IF～THEN文を使って，どの変数でもよいから，−999999が代入されたら，RESTORE するようにしてやるのも一つの手でしょう．

データの行が何行にもわたる場合，RESTOREのあとに行番号を指定すると，指定された行番号のデータからをRESTOREしてくれます．

```
10 DEF FN A(X,Y)=X*Y
20 I=I+1:READ A1,A2,A3,A4
30 Z=FN A(A1,A2):GOSUB 100
40 Z=FN A(A2,A3):GOSUB 100
50 Z=FN A(A3,A4):GOSUB 100
60 DATA 2,3,4,5
70 RESTORE:IF I<2 GOTO 20
80 END
100 IF Z>10 THEN PRINT"ABC"
110 RETURN
RUN
ABC
ABC
ABC
ABC
```

```
40 READ ‥‥
50 RESTORE 1010
60 READ ‥‥‥

1000 DATA 1,2,3,4,5
1010 DATA 6,7,8,9,10
1020 DATA 11,12,13,14,15
```

```
10 CLS
20 READ A,B,C
30 T=A*B*C
40 PRINT "ﾀｲｾｷ ﾊ";
      T;"ﾘｯﾎﾟｳ cm"
50 DATA5,6,7
RUN
ﾀｲｾｷ ﾊ 210 ﾘｯﾎﾟｳ cm
```

```
10 CLS
20 READ A$,B$,C$,C$
30 PRINT A$;B$;C$:PRINT
40 RESTORE
50 READ A$,B$,C$
60 FOR I=1 TO 4:READ D$:NEXT
70 PRINT A$+C$+D$
80 DATA ﾄｳｷｮｳﾄ
90 DATA" ｽｷﾞﾅﾐｸ"," ｾﾀｶﾞﾔｸ"
100 DATA" ｵｷﾞｸﾎﾞ"," ｱｻｶﾞﾔ"
110 DATA" ｶﾐｳﾏ"," ﾊﾞｼﾞｺｳｴﾝ"
RUN
ﾄｳｷｮｳﾄ ｽｷﾞﾅﾐｸ ｵｷﾞｸﾎﾞ

ﾄｳｷｮｳﾄ ｾﾀｶﾞﾔｸ ﾊﾞｼﾞｺｳｴﾝ
```

4.6 TRON, TROFF

プログラムを作っていて，試行錯誤を繰り返しているとき，CRT画面上で実行している動作が，行番号何番で実行しているかがわかれば，たいへん便利なときがあります．TRON，TROFFはそのための命令です．試しに実行する前に，

TRON RETURN

と入れてからRUNしてみましょう．

```
10 DEF FN A(X,Y)=X*Y
20 I=I+1:READ A1,A2,A3,A4
30 Z=FN A(A1,A2):GOSUB 100
40 Z=FN A(A2,A3):GOSUB 100
50 Z=FN A(A3,A4):GOSUB 100
60 DATA 2,3,4,5
70 END
100 IF Z>10 THEN PRINT"ABC"
110 RETURN
RUN
[10][20][30][100][110][40]
                    [100]ABC
[110][50][100]ABC
[110][60][70]
```

[ ]でくくった行番号の表示が出てきます．どのように実行しているかよくわかりますね．この動作を中止させるにはTROFFと入れてください．

◆ UNLIST

LISTと入れてやると，プログラムをCRT上に表示できることはご存知ですね．でも，プログラムを秘密にしておきたいときもあります．こんなときはUNLISTと入れてください．コンピュータはおぼえているプログラムをだれにも教えてくれません．そのかわり，UNLISTのあとではどうしてもLISTを表示してくれないため，プログラム作りをやっている途中では，使わないようにしてください．

◆ RENUM

プログラムを作るとき，行番号を指定しなくてはならず，プログラムの実行は原則として行番号の順であることは，もうご存知ですね．そして，すでに付け終わった行番号の間に，さらに新しい行を追加していくことがあります．こんなとき行番号をもう一度整理して，新しく付け直した方がよい場合が少なくありません．こんな仕事を自動的にやってくれるのが，この命令です．

RENUM[[新行番号][,[旧番号][,増分値]]

かっこの中を省略し，RENUMだけを直接モードで入れてやると，行番号10から10飛びに新しい番号を付けてくれます．10飛びではなく，3飛びにしようと思ったら，増分値を3としてください．旧番号のところは，番号の付けかえを始めるプログラムの行番号であり，新行番号は，その旧番号を今度は何という行番号から始めるのかを示すものです．

RENUMしない以前のプログラムの中にGOTO, GOSUB, THEN, ON～GOTO, ON～GOSUB, ERLなどで参照している行番号があっても，もちろん支障のないように同時に付け直してくれます．

ただし，旧番号を使っている状態の場合，たとえばGOTO 25のとき行番号25にプログラムがない場合には，″Undefined line XXXX in YYYY″（XXXXは存在しなかった行番号，YYYYはその文の新しい行番号）とエラーメッセージが出て，誤った行番号は修正されず，そのまま残ります．

なおRENUMのコマンドでは，行番号64000以上の行番号を発生することはできません．

◆ DELETE

プログラムを作っている途中で，プログラムの一部を行ごとそっくり削除してしまいたい場合が少なくありません．そんなときに必要なのがDELETEです．

> DELETE［行番号1］［{-|,}］［行番号2］］

この働きを例によって調べてみましょう．

DELETE 100−200 RETURN

行番号100～200のプログラムを全て削除してしまいます．なお，−のかわりに ,（コンマ）を使っても結果は同じです（以下同様）．

DELETE 300 RETURN

行番号300のプログラムを削除します．なお，行番号だけをキーインして RETURN キーを押してもその行は削除されてしまいます．

DELETE−400 RETURN

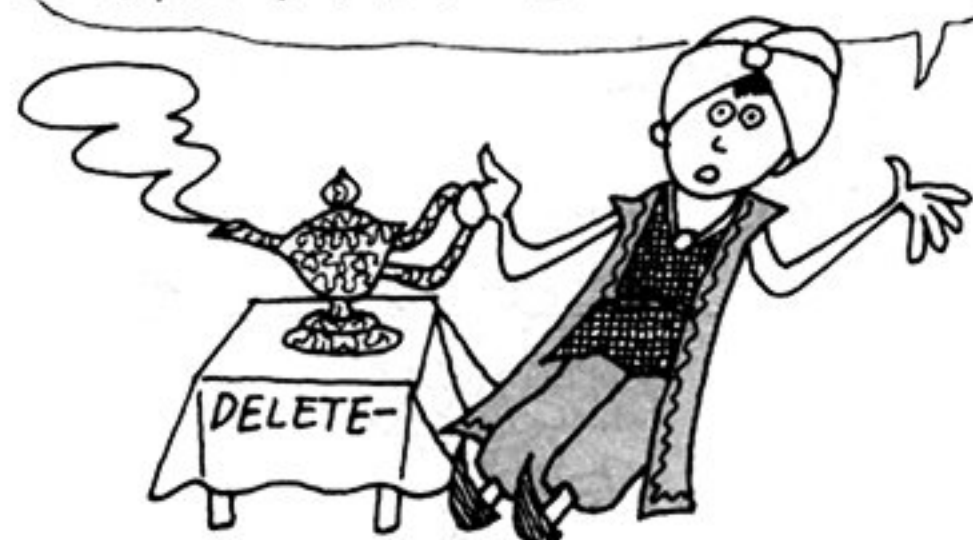
行番号0から400までのプログラムを，そっくり削除してしまいます．

DELETE 500− RETURN

行番号500以後のプログラムを全て削除してしまいます．

DELETE− RETURN

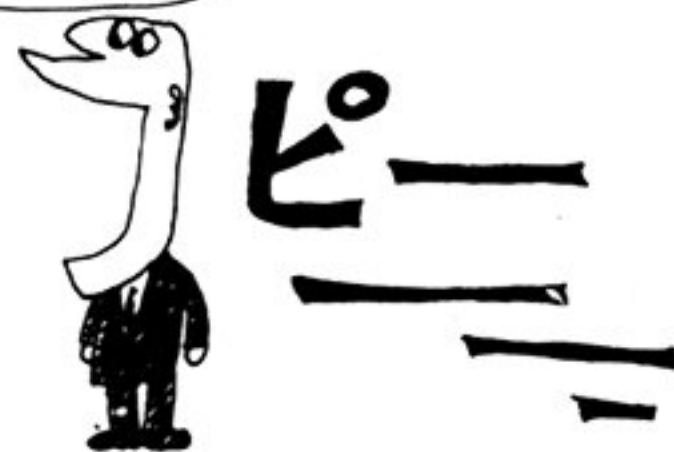
全ての行を削除してしまいます．同じ仕事ははNEW RETURN と入れてやってもできます．

なお，プログラムの中でDELETE文が出てくると，この命令を実行したあと，プログラムを終えてコマンド待ちの状態に入ります．

◆ BEEP

コンピュータに仕事をさせている途中で，人にわかるように音を出させることができます．

> BEEP［スイッチ］

スイッチのところを1とすると，ピーッという音が出始め，BEEP 0に出会うと音は停止します．

```
10 BEEP 1
20 INPUT"ﾓｼﾞ ｦ ｲﾚﾃ ｸﾀﾞｻｲ";A$
30 BEEP 0
40 PRINT A$
```

```
10 BEEP 1
20 FOR I=1 TO 100
30 NEXT
40 BEEP 0
50 FOR I=1 TO 100
60 NEXT
70 GOTO 10
```

ピッピッピッ………

4.7 DIM

変数はデータを入れる箱でしたね．同じ系統の箱をきちんと並べて自由に操作してやりたい場合が少なくありません．そんなとき利用できるのが配列変数です．もちろん配列変数を自由に普通の変数と同じように利用することができるのはいうまでもありません．

DIM　変数名（添字の最大値 ［，添字の最大値］……］）

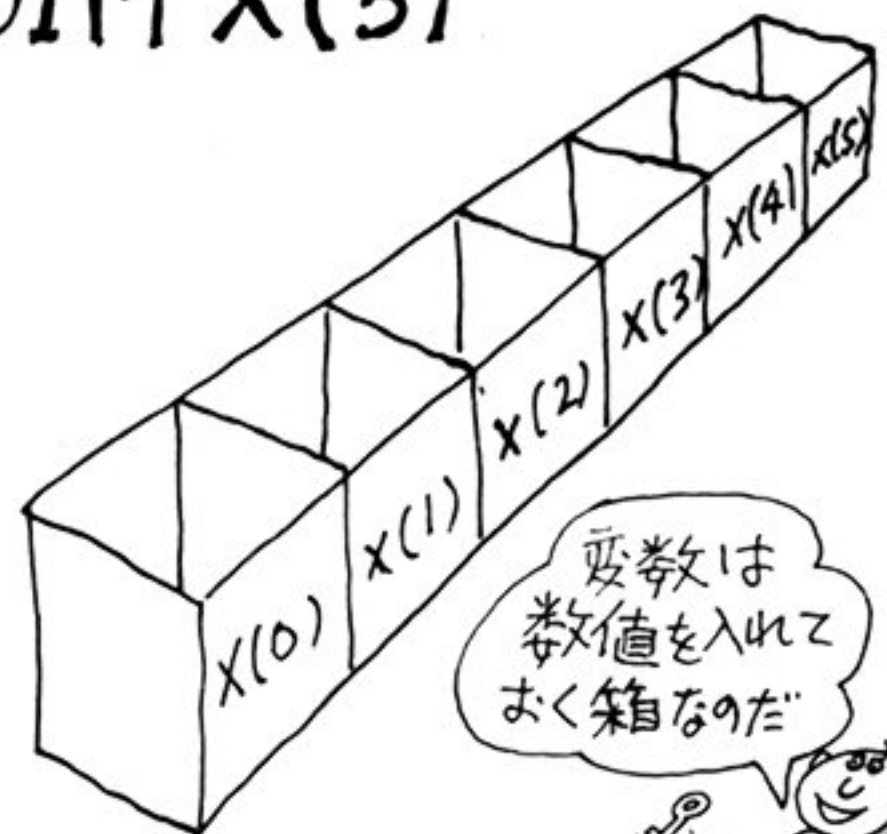
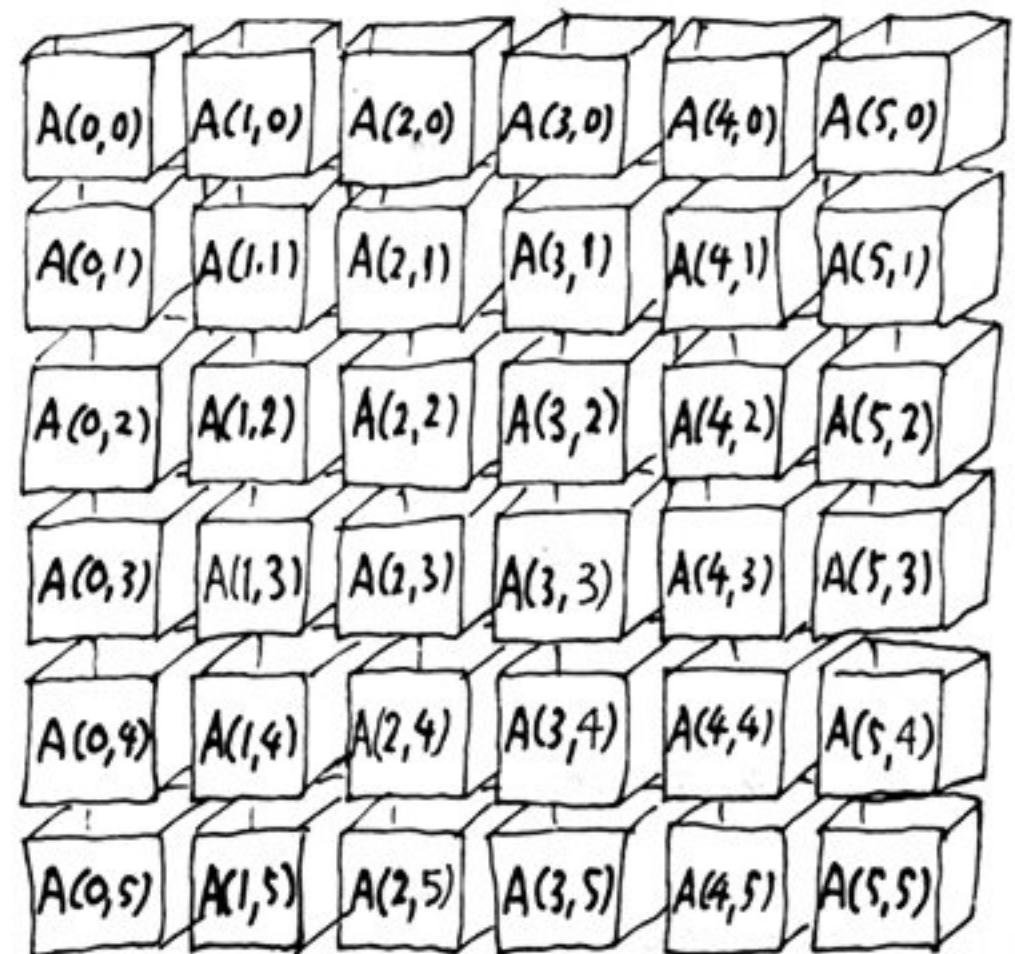
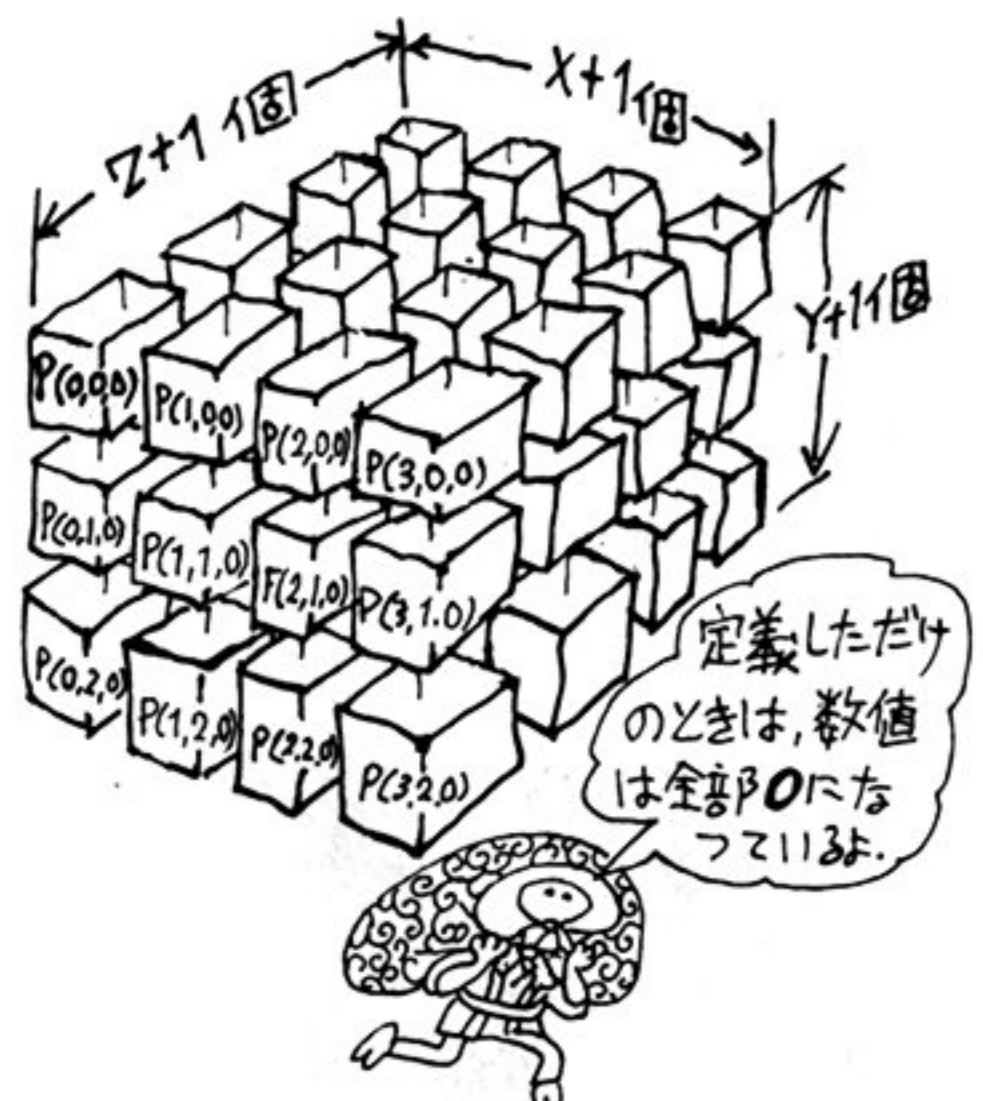
配列変数は，原則として上のような形式で宣言したあとでないと使えません．宣言しないで使うと，添字は自動的に10に設定されます．

DIMA（5,5）とすると，図のようにA（0,0），A(1,0)，A(2,0)……A(5,5）のように合計36個のデータを入れる箱が用意されます．もしDIMA（5）ならA(0)，A(1)，……，A(5)の6個の変数が用意されるわけです．この宣言をしたあと，配列変数を使うと便利な仕事に利用すればよいのです．簡単な例を見てみましょう．

```
10 DIM X(2,2)
20 FOR I=1 TO 2
30 FOR J=1 TO 2
40 X(I,J)=X(I-1,J-1)+2
50 NEXTJ,I
60 FOR J=0 TO 2
70 FOR I=0 TO 2
80 PRINT"X("I","J")="X(I,J)
90 NEXT I,J
```

このプログラムを実行してみると，

```
RUN
X( 0 , 0 )= 0
X( 1 , 0 )= 0
X( 2 , 0 )= 0
X( 0 , 1 )= 0
X( 1 , 1 )= 2
X( 2 , 1 )= 2
X( 0 , 2 )= 0
X( 1 , 2 )= 2
X( 2 , 2 )= 4
```

となります．

8人の学生の成績を，キーインによりコンピュータに伝え，番号順に一覧表にして表示するプログラムです.

```
10 DIM X(8)
20 FOR I=1 TO 8
30 PRINT I"バン ";
40 INPUT X(I)
50 NEXT I
60 CLS
70 PRINT"セイセキ ヒョウ"
80 PRINT
90 PRINT"バンゴウ                セイセキ"
100 FOR J=1 TO 8
110 PRINT J,X(J)
120 NEXT J
```

```
RUN
 1 バン ? 45
 2 バン ? 55
 3 バン ? 90
 4 バン ? 80
 5 バン ? 85
 6 バン ? 100
 7 バン ? 95
 8 バン ? 70
セイセキ ヒョウ
```

バンゴウ	セイセキ
1	45
2	55
3	90
4	80
5	85
6	100
7	95
8	70

本当は行番号10は，添字が10以内なので不要なのだ．11以上なら必要になる．

行番号80

セイセキ ヒョウ

1行あける

バンゴウ	セイセキ
1	45
2	55
3	90

そのためには．プログラムを少しいじくらなくてはならないぞ.

どうすればいいんじゃ

やり方はそのうちあとで教えてくれるんだろう.

数値だけでなくカラー棒グラフで表示できたらいいね．60点以下は赤で表示させたりしてサ.

自分で考えろってか

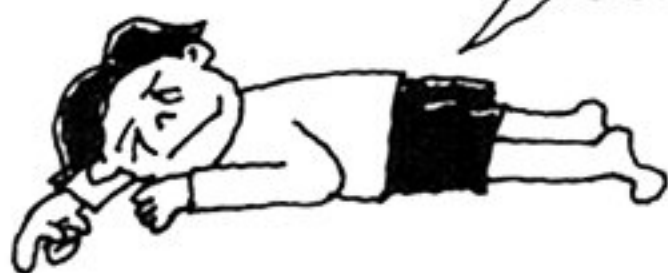
ここではCRT表示しかしていないが，これらのデータを．プリンタに直接印刷し出したり，データとしてファイルに保存し，いろいろと利用することも．できるのである.

なお，人数を８人以下の任意の数のところで中止したいときは，

```
10 WIDTH 40,20
20 DIM X(8)
30 FOR I=1 TO 8
40 PRINT I"バン ";
50 INPUT X(I)
60 IF X(I)>100 THEN 80
70 NEXT I
```

として，100より大きい数を入れたとき，初めのループからぬけ出すようにすればよいのです．

```
80 WIDTH 80,20
90 PRINT"                セイセキ ヒョウ"
100 PRINT
110 PRINT"バンゴウ              セイセキ"
120 IF X(1)>100 THEN PRINT:END
```

```
130 FOR J=1 TO I-1
140 PRINT J,X(J);
150 LOCATE 22,2+J:COLOR 2
160 FOR K=1 TO X(J)¥10
170 IF X(J)<10 GOTO 190
180 PRINT"*";
190 NEXT K
200 PRINT
210 COLOR 7
220 NEXT J
```

```
        セイセキ ヒョウ

バンゴウ          セイセキ
 1                 52        *****
 2                 64        ******
 3                 80        ********
 4                 70        *******
 5                 83        ********
 6                 62        ******
 7                 90        *********
 8                 64        ******
```

ここでは変数X(8)のように，一次元の配列変数と示しましたが，これを学校全体に広げて，

```
DIM X(3,3,40)
```

のように，三次元の配列変数にしてやれば3学年，各学年3クラス，クラスの人数40人までの成績一覧表を作ることができますし，さらに教科別の区別をしたければ四次元配列にすることにより，実行不可能ではありません．

このような仕事は，配列変数を使わないと，とてもやり切れないことがおわかりいただけたと思います．ただし配列変数は，たくさんの箱を用意するわけですから，その分メモリをたくさん使ってしまいます．必要以上の宣言はやめた方がよいでしょう．

◆ たとえば，成績を点数の高いもの順に並べかえる場合のやり方について，基本的なやり方をおぼえておいてください．

```
10 INPUT A,B,C,D
20 IF A<B THEN E=A:A=B:B=E
30 IF B<C THEN E=B:B=C:C=E
40 IF C<D THEN E=C:C=D:D=E
50 IF A<B GOTO 20
60 IF B<C GOTO 20
70 IF A<C GOTO 20
80 PRINT A;B;C;D
```

```
RUN
? 6, -4, 79, 90
 90   79   6  -4
```

行番号20でAよりBの方が大きい場合，AとBの数値を入れかえてやる操作をしています．つまり，まずEという変数の箱を用意します．そして今までAに入っていた値をEに入れ，つぎにBに入っていた値をAに入れます．さらにいったんEに入れておいた値をBに入れてやればよいのです．

◆ 前記のプログラムは，もっと簡単に作ることも可能です．それにはSWAP，つまり交換するという新しい命令が必要です．

SWAP 変数，変数

二つの変数は同じ型でなくてはなりません．

これによって，二つの変数の中身が入れかわってしまいます．これを使ってプログラムを作ってみましょう．

```
10 DIM X(4)
20 FOR I=1 TO 4:INPUT X(I)
   :NEXT
30 FOR I=1 TO 3
40 K=I
50 FOR J=I+1 TO 4
60 IF X(J)>X(K) THEN K=J
70 NEXT J
80 SWAP X(I),X(K)
90 NEXT I
100 FOR I=1 TO 4:PRINT X(I)
    ;:NEXT
110 PRINT
120 GOTO 20
```

行番号20が配列変数にデータを入れる命令で，行番号30～90が数値を交換する作業です．行番号40で，FOR～NEXTの変数Iの中に入っている数値を，Kという別の変数に移し込んでおきます．行番号でX(K)とX(J)とを比較し，結果次第でKをJの値にします．

たとえばI＝1のとき，K＝1です．そしてX(3)＞X(1)だとすると，K＝3ということになるわけです．行番号80で，X(1)の数とX(3)の値を交換してしまいます．変数が4個の場合，30～90のループを3回（4－1回）回してやることにより，全ての数値は大きい順に並んでしまいます．

4.8 ストリング関数

ストリングは糸とか紐(ひも)のことです．コンピュータでは文字列のことを意味します．そして，数値関数や数値定数の場合と同様に，いろいろな加工処理が可能です．それによって，必要に応じたそれぞれ異なる文章や文字を表示してやることができます．

◈ CHR$（式）

普通に表示される文字には，もうご存知のように英文字，カタカナ，数字，各種記号等があります．そして，これらには**付録**に示すように，キャラクタ文字コードという番号が付いています．コードは16進数でいうと，2桁の数字00～FF（10進数で0～255）でできています．

表をごらんください．この2桁の16進数の上位の桁が横，下位の桁が縦に示されています．たとえば上位4，下位Dの4Dの文字は，〝M〟というわけです．このコードによって，文字を表示させることも可能です．CHR$（式）がそれです．式に代入される数値がキャラクタコードになります．$（ドル）印は文字関数を示します．

式に代入される数は10進数でも16進数でもよいのですが，16進数の場合はそのことを示す&H（&はアンパサンドと呼ぶ）を数値の頭に付けなくてはなりません．ではプログラムで文字を表示してみましょう．

```
10 FOR I=&H30 TO &HFF
20 PRINT CHR$(I);
30 NEXT I
```

これを実行すると，図のようになります．なお，ここでは16進数で実行していますが，&H30のかわりに48，&HFFのかわりに255を入れてやっても全く同じ働きをします（48=16×3+0，255=16×15+15）．行番号10の数値を直して試してください．

```
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVW
XYZ[¥]^_`abcdefghijklmnopqrstuvwxyz{|}‾_
▁▂▃▄▅▆▇█▏▎▍▌▋▊▉┼┴┬┤├▔─│▕┌┐└┘╭╮╰╯ 。「」、・ヲァィ
ゥェォャュョッーアイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミ
ムメモヤユヨラリルレロワン゛゜═╞╪╡◢◣◥◤♠♥♦♣●○╱╲╳円年月日時分秒〒
市区町村人▒
```

```
10 FOR I=0 TO 207
20 FOR J=1 TO 70
30 IF 0=I-22*J THEN PRINT
40 NEXT J
50 PRINT CHR$(I+48);
60 NEXT I
```

このプログラムは，一行の文字数を22字に制限するように作り直したものだ．

```
RUN
0123456789:;<=>?@ABCDE
FGHIJKLMNOPQRSTUVWXYZ[
¥]^_`abcdefghijklmnopq
rstuvwxyz{|}‾▁▂▃▄▅▆▇█
▏▎▍▌▋▊▉┼┴┬┤├▔─│▕┌┐└┘╭╮
╰╯ 。「」、・ヲァィゥェォャュョッーアイウ
エオカキクケコサシスセソタチツテトナニヌネノ
ハヒフヘホマミムメモヤユヨラリルレロワン゛゜
═╞╪╡◢◣◥◤♠♥♦♣●○╱╲╳円年月日時
分秒〒市区町村人▒
```

◈ HEX$（式），OCT$（式）

エ？　16進数と10進数との関係がよくわからないですって？　それならコンピュータにおまかせください．試しに，PRINT　&H2FFと入れてみてください．16進数2FFを767と10進数に直して教えてくれます．

10進数を16進数に直すには，HEX$（式）としてください．変換結果は，小数部分を切り捨てたあと実行され，0～FFFFの範囲であなたに教えてくれます．

```
10 INPUT A
20 Z$=HEX$(A)
30 PRINT Z$
40 GOTO 10
RUN
? 20
14
? 100
64
? 200
C8
? 255
FF
```

気をつけていただきたいのは，結果の数は文字列として出てくるので，$印を忘れないようにしていただくことと，この文字列は最大FFFFですから，10進数にして65535までということです．

また，整数を8進数の文字列に変換するには，OCT$（式）を使います．

```
10 PRINT
20 INPUT"10ｼﾝｽｳ ｦ ｲﾚﾖ ";A
30 Z$=OCT$(A)
40 PRINT"8ｼﾝｽｳ ﾃﾞﾊ "Z$
50 GOTO 10
RUN

10ｼﾝｽｳ ｦ ｲﾚﾖ ? 8
8ｼﾝｽｳ ﾃﾞﾊ 10

10ｼﾝｽｳ ｦ ｲﾚﾖ ? 256
8ｼﾝｽｳ ﾃﾞﾊ 400

10ｼﾝｽｳ ｦ ｲﾚﾖ ? 65535
8ｼﾝｽｳ ﾃﾞﾊ 177777

10ｼﾝｽｳ ｦ ｲﾚﾖ ? 65536

Illegal Function Call In 30
```

```
10 WIDTH 80,25:CLS
20 PRINT"10ｼﾝ 8ｼﾝ  16ｼﾝ"
30 FOR I=1 TO 20
40 PRINT I
50 LOCATE 6,I:PRINT OCT$(I)
60 LOCATE 12,I:PRINT HEX$(I)
70 NEXT I
80 LOCATE 20,0
90 PRINT"10ｼﾝ 8ｼﾝ  16ｼﾝ"
100 FOR I=21 TO 40
110 LOCATE 20,I-20:PRINT I
120 LOCATE 26,I-20:PRINT OCT$(I)
130 LOCATE 32,I-20:PRINT HEX$(I)
140 NEXT I
150 LOCATE 40,0
160 PRINT"10ｼﾝ 8ｼﾝ  16ｼﾝ"
170 FOR I=41 TO 60
180 LOCATE 40,I-40:PRINT I
190 LOCATE 46,I-40:PRINT OCT$(I)
200 LOCATE 52,I-40:PRINT HEX$(I)
210 NEXT I
```

◆ 文 字 式

文字列の合成をやってみましょう．文字変数を使う場合，変数名の終わりに＄印を付けることが必要でしたね．文字式は最大255文字のデータを入れることができます．また配列変数の場合には，A$（1），A$（2）……のように書きます．

```
10 A$="ｱｲｳｴｵ"
20 B$="ｶｷｸｹｺ"
30 C$="ｻｼｽｾｿ"
40 D$="ﾀﾁﾂﾃﾄ"
50 Z$=A$+B$+C$+D$
60 PRINT Z$
RUN
ｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿﾀﾁﾂﾃﾄ
```

文字列は，数字の場合のように文字の前に1字分のスペースが出たりはしません．もし出したかったら，″␣アイウ…␣″のように，スペースを取ってください．コンピュータ処理の結果によって，文字の組み合わせを違え，いろいろな文章を表示してやるとおもしろいでしょう．

```
10 A$="ﾌﾙｲｹﾔ "
20 B$="ｶﾜｽﾞ ﾄﾋﾞ ｺﾑ "
30 C$="ﾐｽﾞ ﾉ ｵﾄ"
40 D$=A$+B$+C$
50 PRINT D$
RUN
ﾌﾙｲｹﾔ ｶﾜｽﾞ ﾄﾋﾞ ｺﾑ ﾐｽﾞ ﾉ ｵﾄ
```

```
10 INPUT"5 ﾓｼﾞ ｦ ｲﾚﾖ ";A$
20 A$=A$+" ﾆﾜ ﾆ ｵﾓﾁｬ ﾉ"
30 A$=A$+" ｺﾏ ﾋﾄﾂ"
40 PRINT A$
50 PRINT
60 GOTO 10
RUN
5 ﾓｼﾞ ｦ ｲﾚﾖ ? ﾊﾙｻﾒﾔ
ﾊﾙｻﾒﾔ ﾆﾜ ﾆ ｵﾓﾁｬ ﾉ ｺﾏ ﾋﾄﾂ

5 ﾓｼﾞ ｦ ｲﾚﾖ ? ﾕｳﾀﾞﾁﾔ
ﾕｳﾀﾞﾁﾔ ﾆﾜ ﾆ ｵﾓﾁｬ ﾉ ｺﾏ ﾋﾄﾂ

5 ﾓｼﾞ ｦ ｲﾚﾖ ? ｱｷﾌｶｼ
ｱｷﾌｶｼ ﾆﾜ ﾆ ｵﾓﾁｬ ﾉ ｺﾏ ﾋﾄﾂ

5 ﾓｼﾞ ｦ ｲﾚﾖ ? ﾊﾂﾕｷﾔ
ﾊﾂﾕｷﾔ ﾆﾜ ﾆ ｵﾓﾁｬ ﾉ ｺﾏ ﾋﾄﾂ
```

◆ LEFT$（文字式，文字数）

″アイウエオカキクケ…
1 2 3 4 5 6 7 8 9 ….
LEFT$(Z$,7)

図のように，文字式で並んだ文字列のうち，指定の文字数だけ左（LEFT）側から取り出す働きをします．たとえば前項のZ$は″アイウエオカキクケコサシスセソ″でしたね．そこで，この文字列の中から新しく左から7文字分を取り出し，これを新しい文字列にしてみましょう．

```
10 CLS
20 A$="ｱｲｳｴｵ"
30 B$="ｶｷｸｹｺ"
40 C$="ｻｼｽｾｿ"
50 D$="ﾀﾁﾂﾃﾄ"
60 Z$=A$+B$+C$
70 Z$=LEFT$(Z$,7)
80 PRINT Z$
```

RUN
アイウエオカキ

行番号70では，Z$という変数の箱の中を，60に続けてすぐ書き直していますが，もちろん別の変数名にしてもよいことはいうまでもありません．なお，文字数の部分を全体の長さより大きな数に選ぶと，文字列全体が表示されますし，0だと空文字列，つまり何も文字が出てきません．

また，文字数は変数や式にしてやってもかまいません．この際，ここに入る数字が小数点を含んでいる場合には，四捨五入され整数となります．ここでは，文字式を例としてZ$としましたが，式でよいのですからZ$のかわりにA$+B$+C$にしてももちろんかまわないわけです．

◆ **MID$（文字式，文字位置〔，文字数〕）**
MID$（文字変数名，文字位置〔，文字数〕）
＝文字式

文字列の中から，指定の場所（文字位置）から指定の文字数を取り出す命令です．もし文字数の部分を省略すると，終わりまでの長さになります．

前のプログラムのうち行番号70を，たとえば 70 Z$＝MID$（Z$，6，5）と直し，RUN させてみましょう．結果は，6文字目から5文字分が取り出せるはずです．

```
10 CLS
20 A$="アイウエオ"
30 B$="カキクケコ"
40 C$="サシスセソ"
50 D$="タチツテト"
60 Z$=A$+B$+C$+D$
70 Z$=MID$(Z$,6,5)
80 PRINT Z$

RUN
カキクケコ
```

MID$ の命令は，また別のおもしろい使い方も可能です．

```
10 A$="アイウエオ "
20 A$=A$+"カキクケコ "
30 A$=A$+"サシスセソ "
40 B$="タチツテト "
50 PRINT A$
60 PRINT
70 MID$(A$,7,6)=B$
80 PRINT A$
RUN
アイウエオ カキクケコ サシスセソ

アイウエオ タチツテト サシスセソ
```

A$の中の7文字目から，6文字分，B$に置きかえられてしまう．

MID の中で指定した文字変数 A$ のうち，指定の文字位置から指定の文字数だけ，＝でつなげた文字式（ここでは変数になっている）B$ と置きかえられてしまいます．

```
10 A$="アナタ ノ セイセキ ハ      テン デ゛ス。 コレカラ モ ガ゛ンバ゛ッテ クダ゛サイ。"
20 INPUT"テンスウ ";X
30 B$=STR$(X)
40 IF X>100 THEN END
50 IF X<=100 THEN GOSUB 70
60 GOTO 10
70 MID$(A$,13,4)=B$
80 PRINT
90 PRINT A$
100 PRINT
110 RETURN
RUN
テンスウ ? 70

アナタ ノ セイセキ ハ 70 テン デ゛ス。 コレカラ モ ガ゛ンバ゛ッテ ク
ダ゛サイ。

テンスウ ? 100

アナタ ノ セイセキ ハ 100テン デ゛ス。 コレカラ モ ガ゛ンバ゛ッテ ク
ダ゛サイ。

テンスウ ? 1000

Ready
```

PRINT文で文字変数を合成するときは，＋でも；でもかまいません．

，を使うと間隔をあけて表示されます．

```
10 A$="アイウエオ"
20 B$="カキクケコ"
30 C$="サシスセソ"
40 PRINT A$;B$;C$
50 PRINT A$+B$+C$
RUN
アイウエオカキクケコサシスセソ
アイウエオカキクケコサシスセソ
```

アイウ　カキク
←14字→

上のプログラムで行番号30の **STR$(X)** は数値変数Xを文字変数に変える命令なのだ．

◆ RIGHT$（文字式，文字数）

今度は右端から指定の文字数だけ取り出す命令です．

```
10 A$="アイウエオカキクケコ"
20 B$=RIGHT$(A$,4)
30 PRINT B$
RUN
キクケコ
```

考え方は，今まで出てきた命令と同じです．たとえば，指定の文字数が文字式の文字の長さより大きい場合には，文字式の文字全部となりますし，0なら空文字列となります．

◆ SPACE$（個数）

文字間隔を，指定の文字数だけあけるための命令です．たとえば，SPACE $（5）とすると，5文字分のスペースがとられます．ただし，個数は0から255まででなくてはなりません。

```
10 A$="ABC"
20 B$="XYZ"
30 PRINT A$ SPACE$(5) B$
RUN
ABC      XYZ
```

```
10 A=100
20 B=500
30 PRINT A SPACE$(5) B
RUN
 100       500
```

```
10 A=100
20 B=500
30 PRINT A SPACE$(0) B
RUN
 100  500
```

0+2文字分

◆ STR$（式）

数値を文字列に変える命令です．STR$（式）の式は数値ですが，この命令によって出てきた結果は文字列になっているのです．

```
PRINT STR$(21)
 21
```

同じ21でも，これは数値ではありません．文字列なのです．試してみましょう．

PRINT　3＋ STR＄(21) RETURN

ピーッと音を出し，Type　Mismatchと出てしまいます．もう一つ例を見てみましょう．

```
10 A=10
20 B=5
30 A$=STR$(A)+"+"+STR$(B)+"="
40 PRINT A$;A+B
RUN
 10+ 5= 15
```

◆ VAL（文字式）

STR$とは逆に，文字式を数に変える命令です．プログラムで試してみましょう．

```
10 A$="21"
20 B$="12"
30 C$="&H12"
40 X=VAL(A$)
50 Y=VAL(A$+B$)
60 Z=VAL(C$)
70 PRINT X;Y;Z
RUN
 21  2112  18
```

PRINT Xの数は，文字式A$に入っていた文字が，Yは二つの文字式の結果が，そしてZは16進数であった文字式が数値に直されて出てきました．ただし，最初の文字が数字でない場合（16進数の場合は0～F，8進数の場合は0～7以外の文字）には，結果は0になってしまいます．上のプログラムでいうと，たとえばA$＝″A＝12″であったとすると，X，Yの値は0になります．

◆ STRING$（文字数，キャラクタコードまたは文字式）

同じ文字や記号などを，指定の数だけ書き出せると便利な場合があります．STRING $を使えばそれも簡単にできます．グラフや図表を作るとき便利です．

```
10 INPUT X
20 A$=STRING$(X,42)
30 PRINT A$
40 GOTO 10
```

行番号20の（　）の中に42とあるのは，＊のキャラクタコードです．Xに数値を入れてやると，その数だけ＊を横に並べてくれます．

```
RUN
? 5
*****
? 15
***************
? 30
******************************
```

文字の入る領域は，後述するように300文字分しか初期設定されていないのです．したがって，もっとたくさんの文字を使うときは，たとえば
CLEAR 500
のようにして広げてやる必要があります．

CLEAR 500

300まで

Out of String Space…領域が足りないよ!

STRING$(X,42)の代りに
STRING$(X,"*")でもよい．

STRING$(X,42)で，Xは普通の場合255までです．もしもっと大きな数（ここでは510まで）を入れたければ，ソフトウエアでカバーしてやらなくてはなりません．下のプログラムはその例です．

```
10 CLEAR 510
20 INPUT X
30 IF X>255 THEN Y=X-255:X=255
40 A$=STRING$(X,42)
50 B$=STRING$(Y,42)
60 PRINT A$;B$
RUN
? 500
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
********************
```

255以上の数
X−255
については，B$が引き受ける．

◈ LEN

文字列が何文字になっているかを調べる命令がこれです．

LEN（文字式）

プログラムの例を見ましょう．

```
10 A$="123456789"
20 Z=LEN(A$)
30 PRINT Z
RUN
  9
```

RUNにより，9という結果を得るはずです．ではおなじみの落語，ジュゲムジュゲムが何文字のストリングになっているか調べてみましょう．

```
10 CLEAR 400
20 I1$="ｼﾞｭｹﾞﾑｼﾞｭｹﾞﾑ"
30 I2$="ｺﾞｺｳﾉｽﾘｷﾚ"
40 I3$="ｶｲｼﾞｬﾘｽｲｷﾞｮﾉｽｲｷﾞｮｳﾏﾂｳﾝﾗｲﾏﾂﾌｳﾗｲﾏﾂ"
50 I4$="ｸｳﾈﾙﾄｺﾛﾆｽﾑﾄｺﾛﾔﾌﾞﾗｺｳｼﾞﾉﾌﾞﾗｺｳｼﾞ"
60 I5$="ﾊﾟｲﾎﾟﾊﾟｲﾎﾟﾊﾟｲﾎﾟﾉｼｭｰﾘﾝｶﾞﾝｼｭｰﾘﾝｶﾞﾝﾉｸﾞｰﾘﾝﾀﾞｲ"
70 I6$="ｸﾞｰﾘﾝﾀﾞｲﾉﾎﾟﾝﾎﾟｺﾋﾟｰﾉﾎﾟﾝﾎﾟｺﾅｰ"
80 I7$="ﾉﾁｮｳｷｭｳﾒｲﾉﾁｮｳｽｹ"
90 L=LEN(I1$+I2$+I3$+I4$+I5$+I6$+I7$)
100 PRINT L
```

これをRUNさせると，結果は165と出てくるはずです．一定長さの文字や数字をキーインしてやるプログラムの場合，間違えて余分な数，あるいは足りない文字数を入れてしまうことがありがちです．しかし，LENを使えば簡単に防止することができます．その他，いろいろ工夫して利用できるのがLENを使った命令です．なお行番号10についてはCLEARの項で説明します．

```
10 CLS
20 INPUT "ﾅﾏｴ ｦ ｲﾚﾖ ";N$
30 L=LEN(N$)
40 K=300+L*16+8
50 LINE@(298,96)-(K,110),PSET,4,B
60 LOCATE 19,10:PRINT N$
ﾅﾏｴ ｦ ｲﾚﾖ? FUJITSU
```

FUJITSU

◆ ASC

指定した文字列の最初の文字のキャラクタコードを出してやることができます.

ASC（文字式）

```
10 A$="ABC"
20 Z=ASC(A$)
30 PRINT Z
RUN
 65
```

RUN の結果は 65 です．最初の文字 A のキャラクタコードは，**付録**からわかるように &H41，つまり 10 進数にして 65 だからです．

この文をINPUT文にしてみませんか? 行番号10をINPUT A$ にすると? ときいてきます．そこで ABC と入れてやると，やはり 65 の結果を得ます．

いくつかの文字列
の中から頭文字が
Aのものだけを
選び出すのだ．

```
10 X$(1)="GINZA"
20 X$(2)="AKASAKA"
30 X$(3)="HARAJUKU"
40 X$(4)="UENO"
50 X$(5)="AOYAMA"
60 X$(6)="ASAKUSA"
70 FOR I=1 TO 6
80 A=ASC(X$(I))
90 IF A=65 THEN PRINT X$(I)
100 NEXT I
RUN
AKASAKA
AOYAMA
ASAKUSA
```

```
10 X$(1)="ANDO"
20 X$(2)="ABE"
30 X$(3)="AKIYAMA"
40 FOR I=1 TO 3
50 B=ASC(MID$(X$(I),2,1))
60 IF B=66 THEN PRINT X$(I)
70 NEXT I
RUN
ABE
```

◆ INSTR

指定された文字を探索して，見つかった位置を出す命令です．情報検索などに利用するとたいへん効果を上げることのできる命令ですから，ぜひ大いに利用してください．

**INSTR（[検索開始位置,]
文字式 1，文字式 2）**

例によって簡単なプログラムを見てみましょう．

```
10 A$="TOKYOTO SETAGAYAKU SANGENJYAYA"
20 A=INSTR(2,A$,"S")
30 PRINT MID$(A$,A,10)
RUN
SETAGAYAKU
```

行番号 10 が文字式 1，すなわち A$ です．この中から文字式 "S" が何文字目にあるかを探しているのが行番号 20 です．" " の中が SE であっても SETAGAYAKU であってもかまいませんね．その結果が A です．A には 8 が入っているはずです．つまり，A$ の 8 文字目からという意味です．

そこで，行番号 30 で MID$ を取り出しています．実行させると SETAGAYAKU の 10 文字が

出てくることは，もう説明しなくてもおわかりでしょう．行番号20の（　）の中の2は，A$の2文字目から探せ，という意味です．

```
10 A$="TOKYOTO SETAGAYAKU SANGENJYAYA"
20 A=INSTR(12,A$,"S")
30 PRINT MID$(A$,A,12)
RUN
SANGENJYAYA
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
TOKYOTO SETAGAYAKU

20 21 22 23 24 25 26 27 28 29 30
SANGENJYAYA

行番号20で探し始める文字位置を変えてみました．行番号30も少し違っています．実行の結果はSANGENJYAYAの11字になります．

4.7で示したように，配列変数を使い，たとえばA$（1），A$（2）……に次々にデータを入れてやり，この中から特定の情報だけをぬき出してやるときなどたいへん便利です．

```
10 A$(1)="SATOU TOSHIO"
20 A$(2)="TANAKA HAJIME"
30 A$(3)="SAITOU TETSUO"
40 A$(4)="YANADA TOSHIO"
50 FOR I=1 TO 4
60 A=INSTR(A$(I),"TOSHIO")
70 IF A>0 THEN PRINT A$(I)
80 NEXT I
RUN
SATOU TOSHIO
YANADA TOSHIO
```

◆ SPC，TAB

いずれも，PRINT文の中で，字数をあけてやるときに用いられる命令です．

SPC（個数），TAB（桁位置）

二つの命令を比べてみましょう．

```
10 PRINT"AAA"TAB(10)"A"
20 PRINT"BBB"SPC(10)"B"
RUN
AAA       A
BBB          B
```

行番号10はTAB，20はSPCを示しています．結果は，TABの場合は左から10文字目にAが，そしてSPCの場合はBBBの終わりから10文字目にBが表示されています．

では，（　）の中を100にしてみましょう．

```
10 PRINT"AAA"TAB(100)"A"
20 PRINT"BBB"SPC(100)"B"
RUN
AAA                 A
BBB


                                B
```

この場合，CRT上では横1行40文字になっています．そうです．TABは，1行で表示しきれないときは，1行の文字数で割った余りの値，この場合は100÷40＝2余り20で，左から20文字目にAが表示されているのです．またSPCの方は，ちゃんとBBBのあと100文字目にBの文字が表示されています．これでTABとSPCの類似点，相違点がおわかりいただけたと思います．どちらも（　）の中は最大255です．

TAB，SPCは直接ストリング関数とは関係ありませんが，これらを利用するとき欠くことのできない命令です．

4.9 漢　字

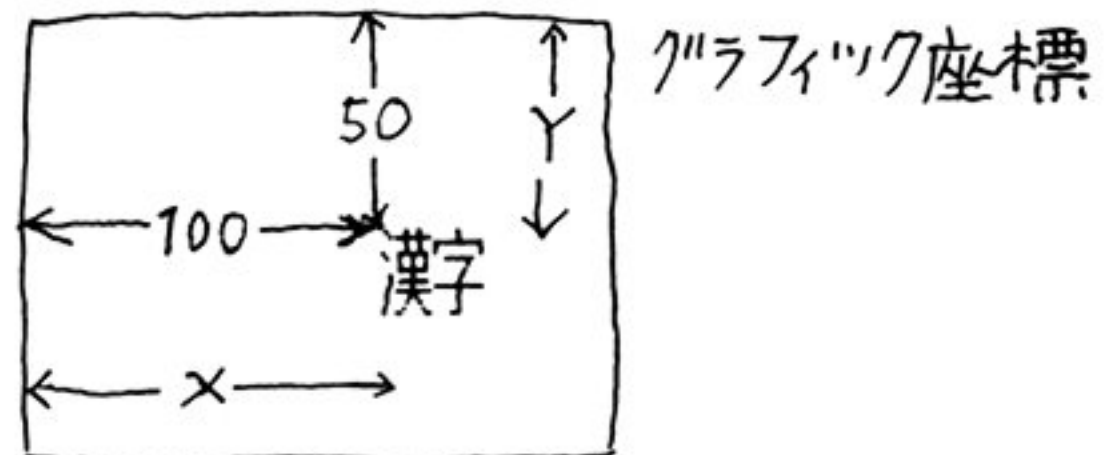
ここでは，付録に示すような漢字やひらがなその他の文字をコードによって指定して，表示する方法を説明します．この方法によって表示できる文字は，漢字2965種，ひらがなその他の各種文字合わせて753種，合計3418種（JIS第一水準）です．このように，漢字だけでなく各種の文字がありますが，ここでは代表して漢字と称することにします．なお，いうまでもないことですが，漢字を表示させるには，パソコンに漢字ROMカードというものを取りつけなくてはなりません．ただし，FM-8は漢字キャラクタセットという16個のROMを装着します．これらの取りつけ方については，説明書を参照してください．

> **PRINT@ [(X, Y),] 漢字コード**
> **[{ , ; }[漢字コード]]……**

はじめのX，Yはグラフィック座標で先頭の文字の位置を示します．つまり，横がX，縦Yの位置に漢字の左上の角がくるわけです．なお，一つの漢字は，16×16のドットパターンでできています．また，座標の指定がないときは，つぎの漢字列は，前に表示された漢字の，つぎの行の左端から表示されます．もちろん，前の漢字の指定の後に，コンマやセミコロンがある場合にはそれにしたがいます．

座標指定の後に続くのが漢字のコードです．付表では，漢字は16ビットつまり16進数で2桁の数値で示されています．これを見ながら『漢字』という文字を画面に表示してみましょう．

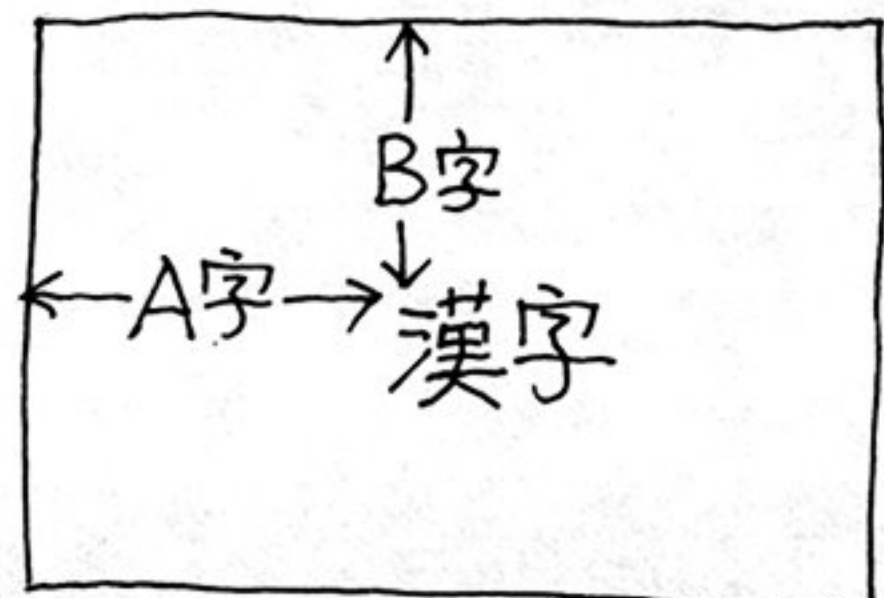
まず，漢という文字です．『カ』の項を見ると344Xの行にあります．そして，横に2文字目ですから，上の数字を見ると1になっています．そこでXのところに1という数をあてはめればよいのです．つまり3441が漢という文字のコードです．もちろん，このコードは16進数ですから，この数の前に&Hという記号をつけなくてはなりませんね．同様にして調べると，字は&H3B7Aということがわかりますね．

```
10 CLS
20 PRINT@(100,50),&H3441,&H3B7A
```

◆ FM-11には，この指定の他に現在のカーソルの位置に漢字を表示させる命令もあります．

> **KANJI 漢字コード[{ , ; }[漢字コード]]…**

実際のプログラムを調べてみましょう．

```
10 CLS
20 LOCATE 10,10
30 KANJI &H3441,&H3B7A
```

これによって，指定の位置から漢字が表示されるはずです．なお，詳細については文法書を参照してください．

LOCATEA, B
KANJI &H3441,&H3B7A

付表から文字を選ぶには……たとえば"年月日"のコードは…

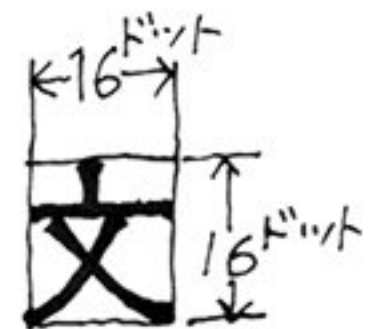
漢字などの日本語を表わすのに，ここではJISのコードを指定しています．富士通には日本語を自由に使い分けできる処理を進めるために，日本語情報処理JEFというたいへんすぐれたシステムがありますが，FMシリーズもこれにしたがっており，JEFコードによって指定することも可能です．

```
10 CLS
20 PRINT@(100,100),&H472F,
   &H376E, &H467C
```

最初の文字の位置の指定　年　月　日

100ドット　100ドット　年月日

JEFコードはJISコード（付表）に，
&H8080
を加えた数だ．
たとえば"ア行の漢字を
&HB0A1～
と指定しても表示できるぞ．

JIS C 622（情報交換用漢字符号系）の漢字表から見る場合は，2進数から16進数に変換しなくてはならない．

富士通パーソナルコンピュータ FM-8 の特長
1. 高分解能，8色カラー表示
640×200ドットで ドット単位の色表示
2. 漢字，ひらがなの表示
3. 音声合成による 音声応答，データの読み上げ
4. バブルカセット，フロッピーディスクにファイルの作成
5. 画面のハードコピー
6. CPU (MBL6809) 2ケ，アドレス空間128KB

FM－8での表示の例

```
10 CLS
20 PRINT@(0,130),12321,
30 FOR I=12322 TO 12361
40 PRINT@ I,
50 NEXT I
```

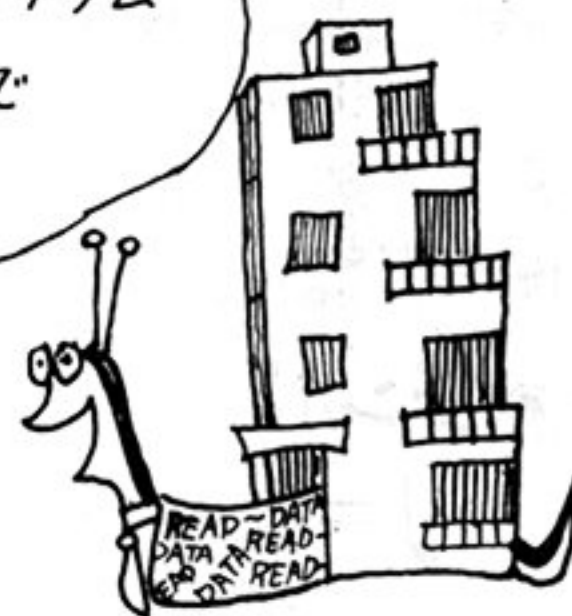
上のプログラムはアで始まる漢字だけを表示するプログラムです．@ のあとの（ ）の数は，CRT 上の座標の位置です．もし（ ）を省略すると，画面左上から順に表示してきます．そして，二回目に RUN すると，前に表示してきた文字の次から表示が始まります．では実行してみましょう．

亜唖娃阿哀愛挨姶逢葵茜穐悪握渥旭葦芦鯵梓圧斡扱宛姐虻飴絢綾鮎或粟袷安庵按暗案闇鞍
杏

次のプログラムはひらがなだけを PRINT します．行番号10は，初めの文字 ″あ″ で位置決めしているわけです．また行番号60は，せっかく表示された文字をこわさないように，カーソルを移してやるための命令です．

```
10 CLS
20 PRINT@(0,100),&H2421,
30 FOR I=&H2422 TO &H2473
40 PRINT@I,
50 NEXT I
60 LOCATE 0,16
```

ぁあぃいぅうぇえぉおかがきぎくぐけげこごさざしじすずせぜそぞただちぢっつづてでと
どなにぬねのはばぱひびぴふぶぷへべぺほぼぽまみむめもゃやゅゆょよらりるれろゎわゐ
ゑをん


```
10 CLS
20 A(0)=&H456C
30 A(1)=&H357E
40 A(2)=&H4554
50 A(3)=&H4024
60 A(4)=&H4544
70 A(5)=&H432B
80 A(6)=&H3668
90 A(7)=&H3B30
100 A(8)=&H382E
110 A(9)=&H4363
120 A(10)=&H3230
130 FOR I=0 TO 10
140 PRINT@(200+20*I,100),A(I)
150 NEXT I
```

東京都世田谷区三軒茶屋

4.10 音楽演奏機能

コンピュータで音楽を奏でてみましょう．

テンポ，音長，音の波形，などを指定する命令があり，これらのコマンドを使って自由に音楽を演奏することができます．なお，ここに示す命令文は，FM-7とFM-11に有効です．

音楽を演奏するためには，音楽の基礎知識が必要ですね．ちょっと復習しておきましょう．

◉ 音程

#A #C #D #F #G #A #C #D #F #G #A #C #D #F #G #A

シ(B) ド(C) レ(D) ミ(E) ファ(F) ソ(G) ラ(A) シ(B) ド(C) レ(D) ミ(E) ファ(F) ソ(G) ラ(A) シ(B) ド(C) レ(D) ミ(E) ファ(F) ソ(G) ラ(A) シ(B)

440Hz(ヘルツ)

◉ 音長

4分音符の半分の長さ

4分音符の2倍の長さ

64分音符　付点64分音符(48分)　32分音符　付点32分音符(24分)　16分音符　付点16分音符(12分)　8分音符　付点8分音符(6分)　4分音符　付点4分音符(3分)　2分音符　付点2分音符　全音符

64分休符　付点64分休符　32分休符　付点32分休符　16分休符　付点16分休符　8分休符　付点8分休符　4分休符　付点4分休符　2分休符　付点2分休符　全休符

◉ テンポ

♩=100 ……… 4分音符(♩)を1分間に100回演奏する速さ．

♩=120 ……… 〃 〃 〃 〃 120 〃 〃 〃

◈ PLAY

音を出すためのコマンドはPLAYです．

```
PLAY"MML"[, "MML"[, "MML"]]
```

MML とは音楽演奏のための言語で，Music Macro Languageの略語です。ここで音の高さや大きさ，長さなどの指定をします。"MML"，を二つまたは三つ並べると和音を出すことができます．

なおFM-11の場合は，和音は出ません。

◈ 音程

前ページのイラストのようにA～Gで音程（音の高さ）をあらわします。半音上げるときのシャープは＃か＋を，半音下げるフラットの場合は，－を音程のあとにつけます．

まず最初に

```
PLAY"C"
```

と入力して RETURN キーを押してください．Cの音が出ましたね．なお，BASIC が起動した初期の状態では，テンポ100で4分音符の長さになっています．

```
10 PLAY"CDEF"
20 PLAY"GFED"
30 PLAY"CDED"
40 PLAY"CEC"
```

◈ オクターブ

オクターブはOであらわし，Oに続く数字でオクターブの上下を示します．1～8までの8オクターブが音域で，初期の状態ではO4に設定されています．

```
PLAY"O5A"
```

上のようにO5とすると，1オクターブ高くなり，O3にすると1オクターブ低くなります．このように数字が大きくなるほど，オクターブが上がるわけです．

なお，楽器の調律などに利用する基準周波数の440Hzの音は，O4のAの音になります．この例では，O5ですから，オクターブ上のAの音つまり880 Hzになります．また，前述のように，PLAY "O5A"を入力した後で

```
PLAY"AGF"
```

とすると，オクターブ上のラ，ソ，ファと続けて演奏されます．これを，初期のオクターブに戻すには

```
PLAY"O4AGF"
```

と，オクターブを指定し直さなくてはなりません．また

のように，ドをオクターブ上にあげる場合は

```
PLAY"O4ABO5C"
```

とします．

◈ 音量

音量はVで指定し，Vのあとに0～15までの数字を入れて音の大きさを変えます．

```
PLAY"V15C"
```

この例のように15にすると一番大きな音が出ます．小さくするときは，この数字を小さくして下さい．なお，初期の設定はV8になっています．

また，オクターブのときと同じく，一度指定したボリュームを変えるときは

```
PLAY"V8CV9EV10G"
```

のように，音程の前でVの指定を入ます．

なお，FM-11では音量の指定はできません．

◈ テンポ

Tはテンポを決めます．つまり，メトロノームの役割をするのがこの命令です．Tに続くのは32～255までの数字で，これは4分音符が1分間に演奏される数にあたります．

```
PLAY"T32C"
```

とすると，4分音符の長さは1分間の32分の1になります．

```
PLAY"T64C"
```

なら64分の1ですから，同じ4分音符でもT32のときの半分の長さになってしまいます．つまり，Tのあとの数値が大きくなるほど，テンポが速くなるわけです．なお，BACICが起動した状態では，T100に設定されています．

◈ 音長

音長はLと1～64の数字であらわします．4分音符はL4にあたります．

```
PLAY"T100L1C"
```

この例のように1のとき最長で，全音符に相当します．したがって2分音符はその半分のL2，8分音符はL8になるわけです．

もちろん1～64までの数値であれば，下表以外の値をとることもできます．

また，付点は音程のあとにピリオドをつけることであらわします．休符はC，D，などの音の指定の代わりにRを入れればいいのです．

```
PLAY"L4CDERE4.D4.C4."
```

なお，上の例のように，L4C L8Aなどとする代わりに，C4 A8とすることもできます．この場合，付点音符は，C4.A8.のように，音長の数字の後に・を入れてください．

それでは，簡単なメロディをならしてみましょう．

```
10 PLAY"V8T100O4L4"
20 PLAY"CDEFGABO5C"
```

ドレミファ～の音階のプログラムです．行番号10で，V，T，L，Oを初期の設定にもどしています．20でドレミファ～を演奏しました．20の最後のCの音はO5としてオクターブを上げているのが分かりますね．

なお，FM-11の場合は，行番号10のV8の指定はできません．

音符の指定

	全音符	L1	L1A*		付点16分音符	L12	L16A.*
	付点2分音符	L2L4	L2A.		16分音符	L16	L16A
	2分音符	L2	L2A		付点32分音符	L24	L32A.
	付点4分音符	L3	L4A.		32分音符	L32	L32A
	4分音符	L4	L4A		付点64分音符	L48	L64A.
	付点8分音符	L6	L8A.		64分音符	L64	L64A
	8分音符	L8	L8A				

*はAの音程の例を示す．

休符の指定

	全休符	R1		付点16分休符	R12
	付点2分休符	R2R4		16分休符	R16
	8分休符	R2		付点32分休符	R24
	付点4分休符	R3		32分休符	R32
	8分休符	R4		付点64分休符	R48
	付点8分休符	R6		64分休符	R64
	8分休符	R8			

◆ 和音

PLAY 文を使って和音を出してみましょう．FM-7は，3個の音源を持っているため，3重和音を出すことができます．

```
PLAY"C","E","G"
```

```
10 PLAY"V8T60O4L4",
     "V8T60O4L4",
     "V8T60O4L4"
20 FOR I=1 TO 2
30 PLAY"D","F","O4A"
40 PLAY"D","F","B"
50 PLAY"E","G","O5C"
60 PLAY"R","R","R"
70 NEXT I
```

このような和音の機能を応用して作ったのが，次のプログラムです．なお，行番号160のSとMについては，後述のエンベロープパターンの項を参照してください．

```
10 CLEAR 800
20 A1$="O4E4C4R4E4C4R4E4G8R
       8G8E8C8E8"
30 A2$="D4R4F4O3B4R4O4D4"
40 A3$="O3G4R4O4D4F8R8F8D8O
       3B8O4D8C4R4"
50 B1$="O3C4E8R8E8R8"
60 B2$="O3F4G8R8G8R8"
70 B3$="O2B4O3D8R8F8R8"
80 B4$="O2B4O3F8R8F8R8C4R4"
90 C1$="R4O3C8R8C8R8"
100 C2$="O2R4B8R8B8R8"
110 C3$="O2R4B8R8O3D8R8"
120 C4$="O3R4D8R8D8R8E4R4"
130 X1$=A1$+A2$+A3$
140 X2$="R4"+B1$+B1$+B1$+
       B2$+B2$+B3$+B4$
150 X3$="R4"+C1$+C1$+C1$+
       C2$+C2$+C3$+C4$
160 PLAY "T130S0M3500"+X1$
170 PLAY "V12T130"+X1$,"V9T
       130"+X2$,"V9T130"+X3$
```

◆ 特殊音程

音程をNと1～96の数字で指定する方法もあります．この方法では，音程を半音も含めて12音程として扱います．たとえば，O1CはN1，O1C# はN2といった具合に，O8のBまでを1～96の数字に置きかえることができます．これを特殊音程といい，音程との対応は下表のとおりです．

```
PLAY"N46"
```

とした場合，N46は表からわかるようにO4のAですから

```
PLAY"O4A"
```

音程と特殊音程(Nn)の対応表

音	程 ＼ オクターブ	O1	O2	O3	O4	O5	O6	O7	O8
シ	B	N12	N24	N36	N48	N60	N72	N84	N96
	A# A+ B−	N11	N23	N35	N47	N59	N71	N83	N95
ラ	A	N10	N22	N34	N46	N58	N70	N82	N94
	G# G+ A−	N9	N21	N33	N45	N57	N69	N81	N93
ソ	G	N8	N20	N32	N44	N56	N68	N80	N92
	F# F+ G−	N7	N19	N31	N43	N55	N67	N79	N91
ファ	F	N6	N18	N30	N42	N54	N66	N78	N90
ミ	F	N5	N17	N29	N41	N53	N65	N77	N89
	D# D+ E−	N4	N16	N28	N40	N52	N64	N76	N88
レ	D	N3	N15	N27	N39	N51	N63	N75	N87
	C# C+ D−	N2	N14	N26	N38	N50	N62	N74	N86
ド	C	N1	N13	N25	N37	N49	N61	N73	N85

N46＝O4A＝440Hz

と同じ意味になります．

この命令の使い方で輪唱というのはいかがでしょう．

```
10 PLAY"T130L4V8",
   "T130L4V8","T130L4V8"
20 A$(3)="N37N39N41N42N41
   N39N37R"
30 A$(4)="N41N42N44N46N44
   N42N41R"
40 A$(5)="N37R4N37R4N37R4
   N37R"
50 A$(6)="N37N39N41N42N41
   N39N37R"
60 FOR I=1 TO 6
70 PLAY A$(I),A$(I+1),
   A$(I+2)
80 NEXT I
```

Nの後の数値を，プログラムによって変化させることもできます．

```
10 FOR I=37 TO 49
20 PLAY"N"+STR$(I)
30 NEXT I
```

◆ エンベロープパターン

音の立ち上がり方や波形などを調整し，音色を変えてやるのがエンベロープパターンです．これを指定するには，SとMの命令を使います．Sは0～14で指定し，これによって以下の表のパターンを選択することができます．なお，エンベロープパターンはFM-7のみに有効です．

パターン0(ゼロ)（0～3，9のいずれか）の波形を使って，ピアノタッチの音を作ってみましょう．

```
PLAY"S0M2000CDEFG"
```

これをオルガンの音に近づけるには，0を13に変えてやればいいのです．Mの指定については次のエンベロープ周波数の項をごらんください．

なお，Sで指定した場合は，Vコマンドによって指定した数値は無効となります．Sを解除するには，再びVコマンドで音量を指定してください．

◆ エンベロープ周波数

MはSとペアで使用します．Mの後に0～65535までの数字をつけて，Sで指定したパターンの周期を指定してやります．つまり，下の表でいうと，横軸の時間を表すTを指定するのがこの命令です．

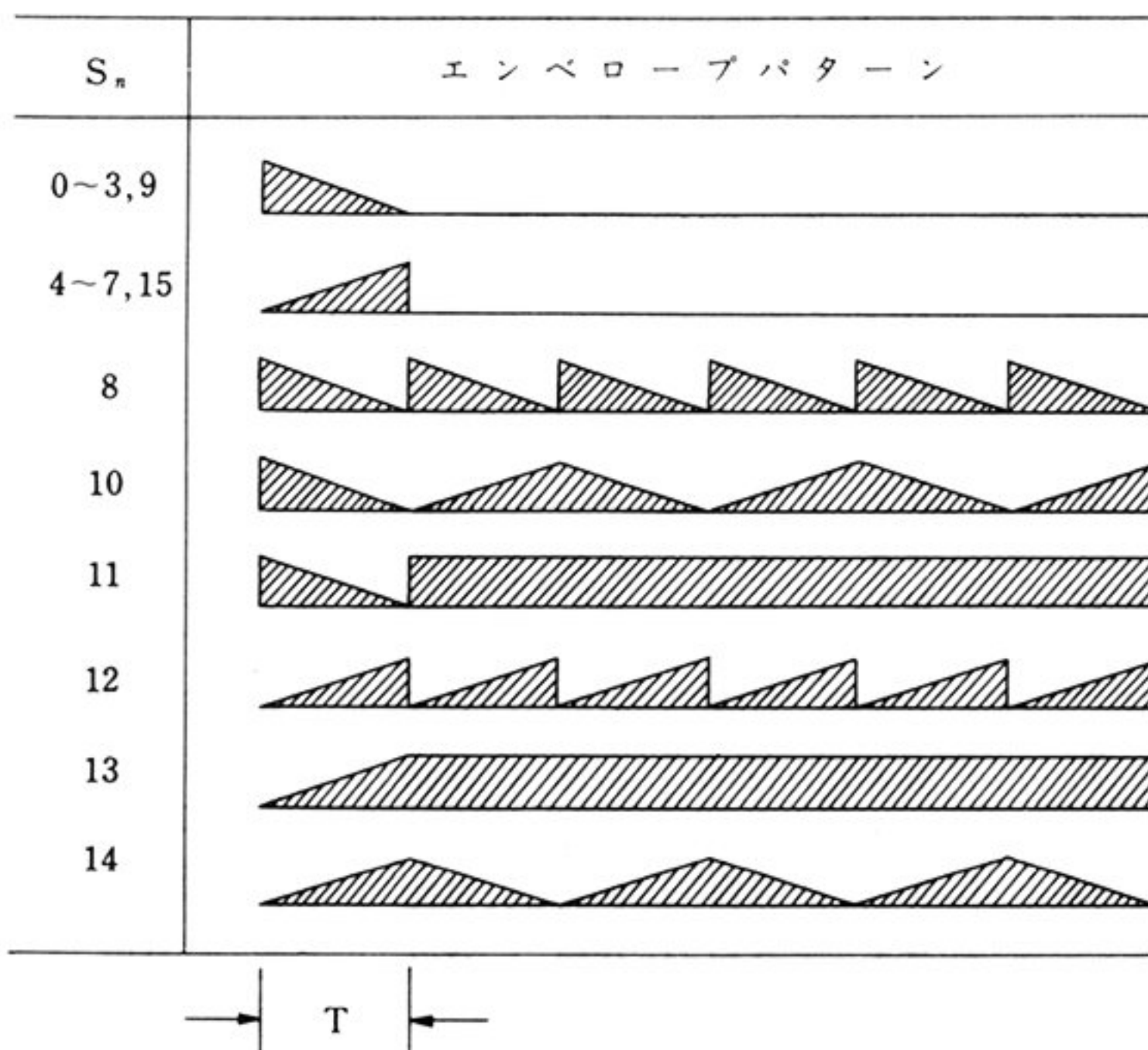
以下にTの計算式を示します．

$$T=\frac{256\times n}{f}\ (\text{秒})\qquad n=1\sim 65535$$

$$=\frac{2.08\times n}{10^4}\quad \text{ただし，}f=1.2288\text{MHz}$$

エンベロープパターンとSコマンドの数値nとの対応表
（縦軸は音量，横軸は時間）

S_n	エンベロープパターン
0～3,9	
4～7,15	
8	
10	
11	
12	
13	
14	

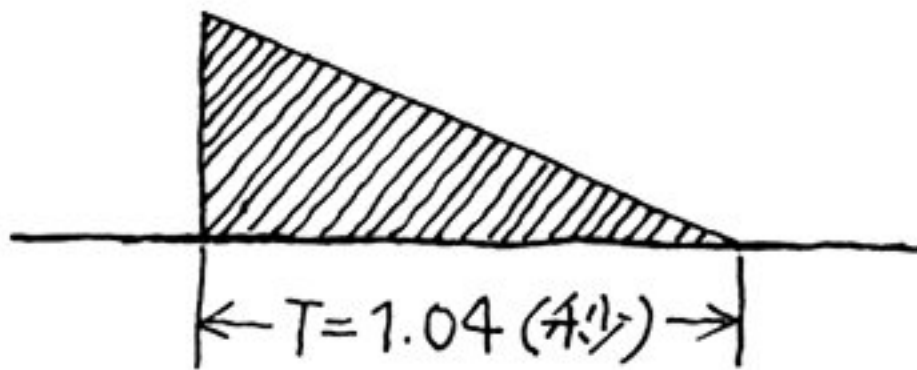
→| T |←

この式から，TはMの後の数値に比例して大きくなっていくのがわかります．簡単な例で調べてみましょう．

```
PLAY"S0M5000A"
```

M5000を前ページの式で計算すると

$$T=\frac{2.08\times n}{10^4}=\frac{2.08\times 5000}{10^4}$$

$$=1.04$$

つまり，1.04秒ということになります．したがって0のパターンの波形は1.04秒で演奏されることになります．

```
PLAY"S8M1000T32L1C"
```

この例では，8の波形の一つ一つが0.208秒で演奏されます．ここでは，波形を分りやすくするために，音を長く設定してあります．この例のSとMの数値を入れかえていろいろと試してください．

SとMの命令を使って，コンピュータのキーボードをピアノのキーに仕立てたのが，下に示すプログラムです．たたいた音が画面上に表示される楽しいプログラムです．

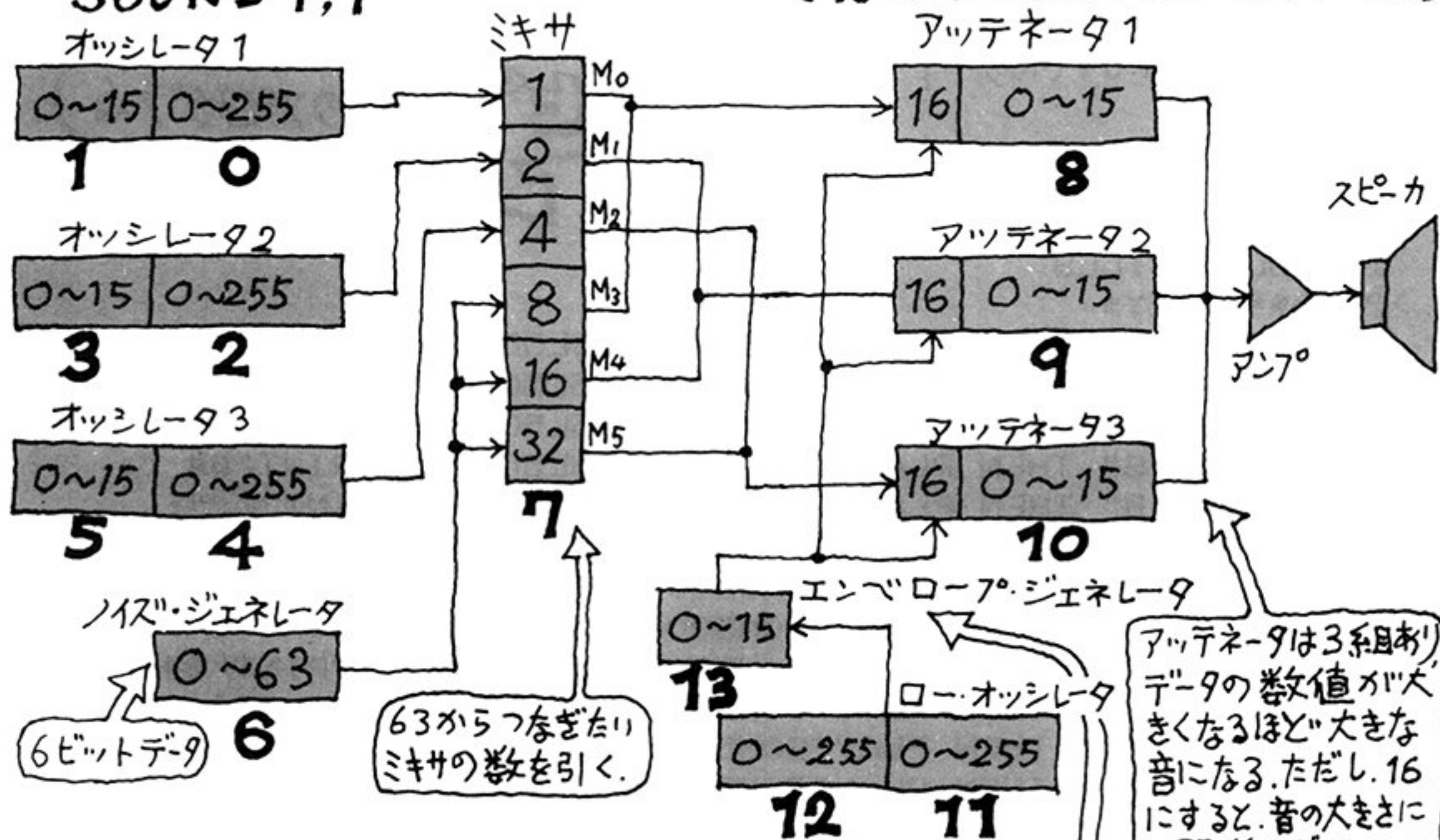
```
10 COLOR 4:CLS
20 A$=INKEY$
30 IF A$="C" THEN PLAY"S0M5000T255L5O4C":PRINT"C ";
40 IF A$="F" THEN PLAY"S0M5000T255L5O4C+":PRINT"C# ";
50 IF A$="V" THEN PLAY"S0M5000T255L5O4D":PRINT"D ";
60 IF A$="G" THEN PLAY"S0M5000T255L5O4D+":PRINT"D# ";
70 IF A$="B" THEN PLAY"S0M5000T255L5O4E":PRINT"E ";
80 IF A$="N" THEN PLAY"S0M5000T255L5O4F":PRINT"F ";
90 IF A$="J" THEN PLAY"S0M5000T255L5O4F+":PRINT"F# ";
100 IF A$="M" THEN PLAY"S0M5000T255L5O4G":PRINT"G ";
110 IF A$="K" THEN PLAY"S0M5000T255L5O4G+":PRINT"G# ";
120 IF A$="," THEN PLAY"S0M5000T255L5O4A":PRINT"A ";
130 IF A$="L" THEN PLAY"S0M5000T255L5O4A+":PRINT"A# ";
140 IF A$="." THEN PLAY"S0M5000T255L5O4B":PRINT"B ";
150 IF A$="/" THEN PLAY"S0M5000T255L5O5C":PRINT"C ";
160 IF A$="R" THEN PLAY"S0M1000T255L5O5C":PRINT"C ";
170 IF A$="5" THEN PLAY"S0M1000T255L5O5C+":PRINT"C# ";
180 IF A$="T" THEN PLAY"S0M1000T255L5O5D":PRINT"D ";
190 IF A$="6" THEN PLAY"S0M1000T255L5O5D+":PRINT"D# ";
200 IF A$="Y" THEN PLAY"S0M1000T255L5O5E":PRINT"E ";
210 IF A$="U" THEN PLAY"S0M1000T255L5O5F":PRINT"F ";
220 IF A$="8" THEN PLAY"S0M1000T255L5O5F+":PRINT"F# ";
230 IF A$="I" THEN PLAY"S0M1000T255L5O5G":PRINT"G ";
240 IF A$="9" THEN PLAY"S0M1000T255L5O5G+":PRINT"G# ";
250 IF A$="0" THEN PLAY"S0M1000T255L5O5A":PRINT"A ";
260 IF A$="O" THEN PLAY"S0M1000T255L5O5A+":PRINT"A# ";
270 IF A$="P" THEN PLAY"S0M1000T255L5O5B":PRINT"B ";
280 IF A$="@" THEN PLAY"S0M1000T255L5O6C":PRINT"C ";
290 GOTO 20
```

SOUND PSGのレジスタ番号，データ

PSGとは，Programable Sound Generatorの略語で，FM－7の発振装置のことです．SOUND命令を使うときは，PSGのしくみを理解する必要があります．下はPSGの構造図です．なお，この命令はFM－7のみに適用できます．

◉太字の数字がレジスタ番号．たとえば **SOUND7,D** とすると，ミキサに対して，**D** というデータを指令することになる．

◉ミキサには６個のスイッチがあり，データによってどのスイッチが開くかが決まる．たとえば，M_0，M_2，M_3をつなぎたいときは 63－(1+4+8)=50 から **SOUND 7,50**にする．

◉オッシレータは３組あり，それぞれ２組のレジスタからできている．データは16進数にして３けたの値まで指定できる．この３けたの数のとき，上位１けたは左側のレジスタ，下位２けたの数は右側のレジスタに入れる．たとえば **D**が**&H15D**なら

PSGの構造図

◉オッシレータの発振周波数はつぎの式による．

$$f=76800\times\frac{1}{D}\ (\mathrm{Hz})$$

音程とDとの関係は次ページの表参照．

◉ノイズ・ジェネレータの平均周波数も上式による．

◉ロー・オッシレータはエンベロープの長さ **T** を決める．

$$T=2.08\times10^{-4}\times D\ (\text{秒})$$

Dの値は，16進数にして４けた（16ビット）の数．この数の上位２けた（8ビット）はレジスタ**12**，下位２けた（8ビット）はレジスタ**11**に入る．

最初にオッシレータとアッテネータを使って音を出してみましょう．

```
10 SOUND 0,175
20 SOUND 8,8
30 SOUND 7,62
```

下表から440Hzは &HAF 10進数に直すと 175 になる

行番号10でレジスタ0をセットしています．データは175ですから，440Hzの音がでることになります．行番号20はレジスタ0と対応するアッテネータのレジスタ8を設定して，音量を8に指定しています．

```
10 SOUND 0,175
20 SOUND 8,8
30 SOUND 7,54
```

この状態では，ミキサの1と8が開いていて，オッシレータからの音とノイズジェネレータからのノイズがいっしょに聞こえてきます．なお，ノイズをカットするには，行番号30行のデータを62にすればいいのです．

```
10 FOR I=1 TO 255
20 SOUND 0,I
30 SOUND 8,12
40 NEXT I
50 SOUND 7,63
```

この例のように，データに変数を使うこともできます．また，音を止めるときの一例として，行番号50でミキサのチャンネルを全部閉じています．

次のプログラムは，ノイズジェネレータとエンベロープを使った例です．

```
10 SOUND 7,55
20 SOUND 8,16
30 SOUND 6,30
40 SOUND 11,0
50 SOUND 12,7
60 FOR I=1 TO 80
70 SOUND 13,4
80 IF I>40 GOTO 100
90 FOR J=1 TO 16*(40-I)
     :NEXT J
100 FOR J=1 TO 16*(I-40)
     :NEXT J
110 NEXT I
```

◆ SOUND 周波数，持続時間

この命令は，FM-11のみに適用できます．

```
SOUND 440,200
```

この例の440が周波数，200が持続時間となります．周波数は20～32767Hz，持続時間は，0.05秒単位で1～65535倍の長さを指定できます．たとえば，880Hzの音を10秒間ならす場合は，

10÷0.05＝200ですから．

```
SOUND 880,200
```

とすればいいのです．

音程(周波数)とデータ(D)の関係(データは16進数)

	O1		O2		O3		O4		O5		O6		O7		O8	
	周波数	データ	周波数	データ	周波数	データ	周波数	データ	周波数	データ	周波数	データ	周波数	データ	周波数	データ
C	32.703	92C	65.406	496	130.81	24B	261.63	126	523.25	93	1046.5	49	2093	25	4186	12
C♯	34.648	8A9	69.296	454	138.59	22A	277.18	115	554.37	8B	1108.7	45	2217.5	23	4434.9	11
D	36.708	82C	73.416	416	146.83	20B	293.66	106	587.33	83	1174.7	41	2349.3	21	4698.6	10
D♯	38.891	7B7	77.782	3DB	155.56	1EE	311.13	F7	622.25	7B	1244.5	3E	2489	1F	4978	F
E	41.203	748	82.407	3A4	164.81	1D2	329.63	E9	659.26	74	1318.5	3A	2637	1D	5274	E
F	43.654	6DF	87.307	370	174.61	1B8	349.23	DC	698.46	6E	1396.9	37	2793.8	1B	5587.7	E
F♯	46.249	67D	92.499	33E	185	19F	369.99	D0	739.99	68	1480	34	2960	1A	5919.9	D
G	48.999	61F	97.999	310	196	188	392	C4	783.99	62	1568	31	3136	18	6271.9	C
G♯	51.913	5C7	103.83	2E4	207.65	172	415.3	B9	830.61	5C	1661.2	2E	3322.4	17	6644.9	B
A	55	574	110	2BA	220	15D	440	AF	880	57	1760	2C	3520	16	7040	B
A♯	58.2 7	526	116.54	293	233.08	14A	466.16	A5	932.33	52	1864.7	29	3729.3	15	7458.6	A
B	61.735	4DC	123.47	26E	246.94	137	493.88	9C	987.77	4E	1975.5	27	3951.1	13	7902.1	9

4.11 TIME$, DATE$, TIME, DATE.

FMシリーズには時間をきざんでいく時計が内蔵されています.

TIME$

TIME$ の形式は,

HH : MM : SS です.

HH……00～23 の時間

MM……00～59 の分

SS……00～59 の秒

であり, 電源投入直後は, 全て 00 : 00 : 00 にセットされています.

同様に内蔵タイマは, 日付を表わすこともできます.

DATE$

DATE の型式は,

YY/MM/DD

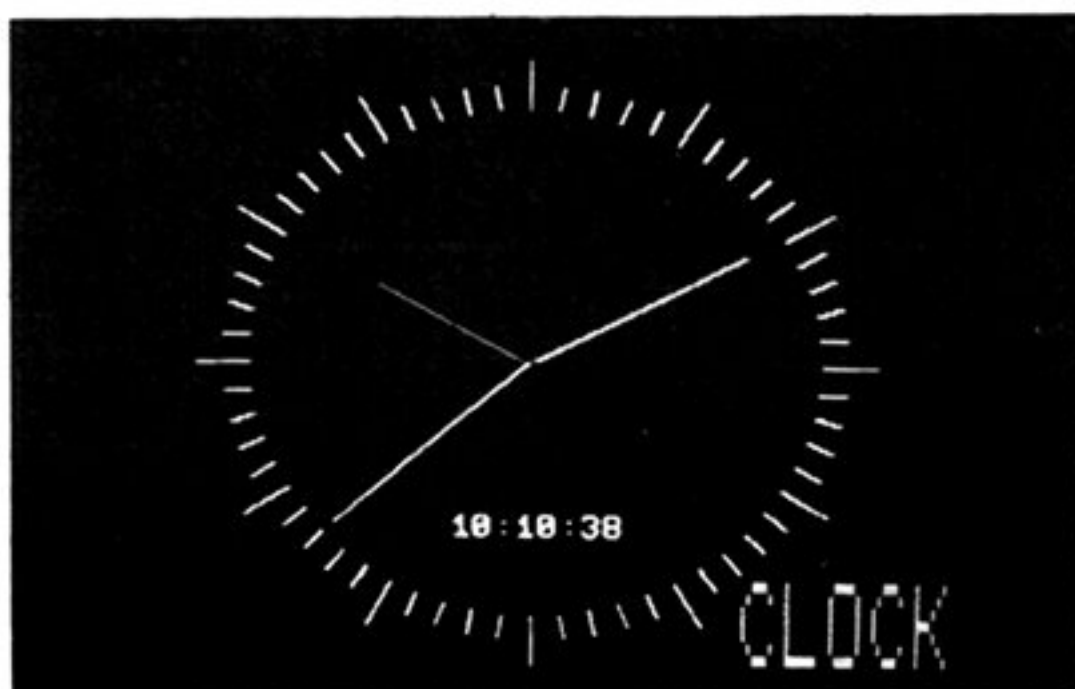
です.

YY……西暦による年で下 2 桁

MM……月

DD……日

で示されます.

どのような使い方をするかはプログラム例を見てください.

```
10 TIME$="23:59:30"
20 DATE$="84/02/28"
30 CLS
40 COLOR 4,0
50 LOCATE 15,8,1
60 PRINT DATE$
70 COLOR 7,0
80 LOCATE 15,10,1
90 PRINT TIME$
100 GOTO 40
```

行番号 10, 20 が時間と年月日の初期値の設定です. 1984 年 2 月 28 日の 23 時 59 分 30 秒に設定されています.

RUN すると, 画面をクリヤしたあと, 年月日と時間を表示してきます. 1984 年は, うるう年ですから, 2 月は 29 日まであります. 正しく動作するかどうかを, 実際に試してみてください.

```
84/02/28          84/02/29
          ⇨
23:59:30          00:00:01
```

行番号 20 で年を 85 年にすると,

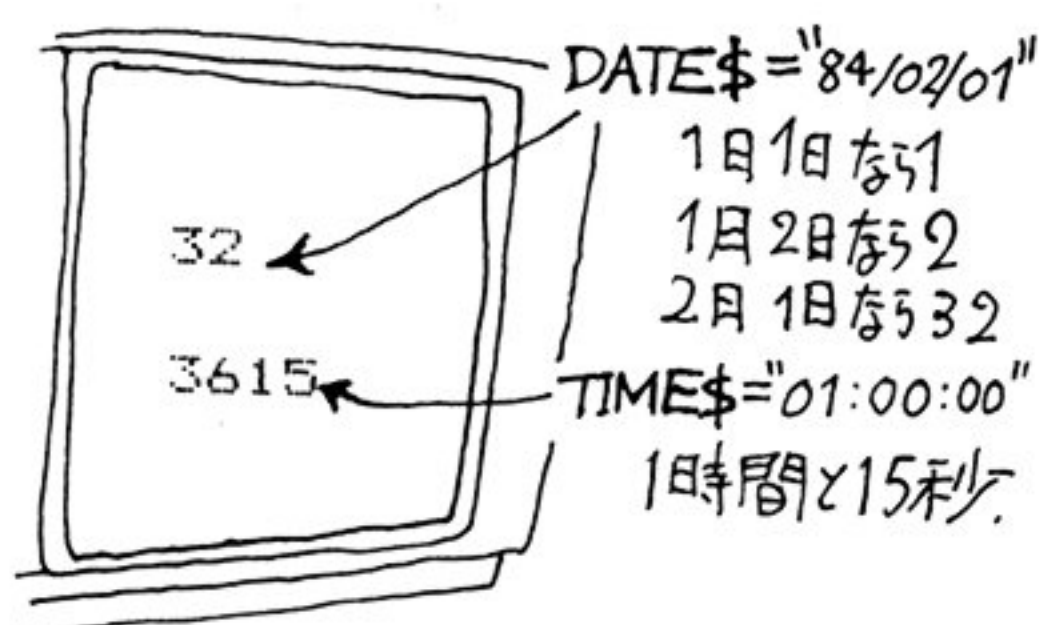
```
10 TIME$="23:59:30"
20 DATE$="85/02/28"
30 CLS
40 COLOR 4,0
50 LOCATE 15,8,1
60 PRINT DATE$
70 COLOR 7,0
80 LOCATE 15,10,1
90 PRINT TIME$
100 GOTO 40
```

```
85/02/28          85/03/01
          ⇨
23:59:30          00:00:01
```

いろいろと工夫して時計のプログラムを作ってみませんか? 左図のような時計を画面に表示させたり, 時間になるとアラーム音をだしたり, 音楽をかなでたりする, 楽しい時計を作ることも可能なはずです. 音楽の演奏方法は前述のとおりです. アラームを中止するには, どのキーを押してもよいように作るのがいいでしょう. 特に目覚し時計の場合間違い易いからです.

似たような使い方ができる命令として，TIME と DATE というシステム変数があります．TIME は，00：00：00 から秒単位の時刻を変数 TIME の中に入れ込みます．また DATE は，1 月 1 日を基準とする合計日数を入れ込みます．

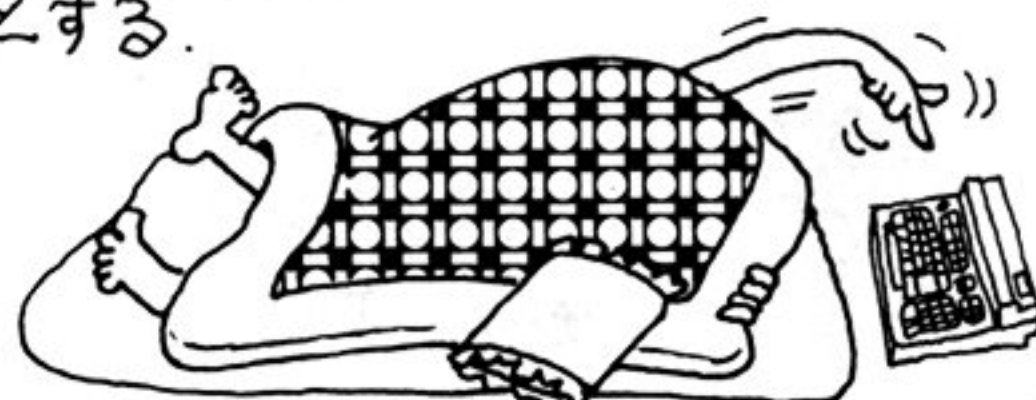
TIME，DATE のプログラム例を示します．

```
10 TIME$="01:00:00"
20 DATE$="84/02/01"
30 CLS
40 COLOR 4,0
50 LOCATE 15,8,1
60 PRINT DATE
70 COLOR 7,0
80 LOCATE 15,10,1
90 PRINT TIME
100 GOTO 40
```

行番号 20 で 84 年の 2 月 1 日，行番号 10 で 1 時間を設定しているため，このプログラムを実行すると，上側に緑の文字で 1 月 1 日からの日数を，下側に白で 0 秒から何秒たっているかを表示してきます．TIME$ が 1 時間，つまり 3600 秒経た後ですから，3600 以上の数が 1 秒ごとに 1 ずつ加えられていきます．

このような時間の経過を数えていく仕事と，たとえばある時間ごとに何らかのデータ（たとえば計測器のデータ読み取りなど）を読み取ったりするときに便利です．

```
10 CLS:T=5
20 TIME$="00:00:00"
30 IF TIME=T THEN BEEP 1
40 IF TIME=T+2 THEN BEEP 0
50 COLOR 4,0:LOCATE 15,10,1
60 PRINT TIME$
70 GOTO 30
```

```
10 INPUT"ｼﾞｺｸｦｲﾚﾖ HH:MM:SS "
   ;T$
20 TIME$=T$
30 INPUT"ｱﾗｰﾑｼﾞｺｸ HH:MM:SS "
   ;A$
40 CLS
50 LOCATE 15,10,1
60 IF TIME$=A$ THEN BEEP 1:
   INPUT"9 ｦｲﾚﾖ";A
70 IF A=9 THEN BEEP 0:A=0:
   CLS:PRINT"ｱﾗｰﾑｽﾄｯﾌﾟ"
80 PRINT TIME$
90 GOTO 50
```

現在の時刻とアラーム時刻を入れる．
"08:30:00" RETURN ⇦現在の時刻
"12:40:00" RETURN ⇦アラーム時刻
（" "でかこむことに注意）
アラーム時刻になるとピーがなる．
ピーをとめるには
9 RETURN
とする．

4.12 タイマ割り込み

◆ ON TIME GOSUB

内蔵時計の時刻が，指定した時刻になった時点で，割り込みをするサブルーチンであり，形式は次のとおりです．

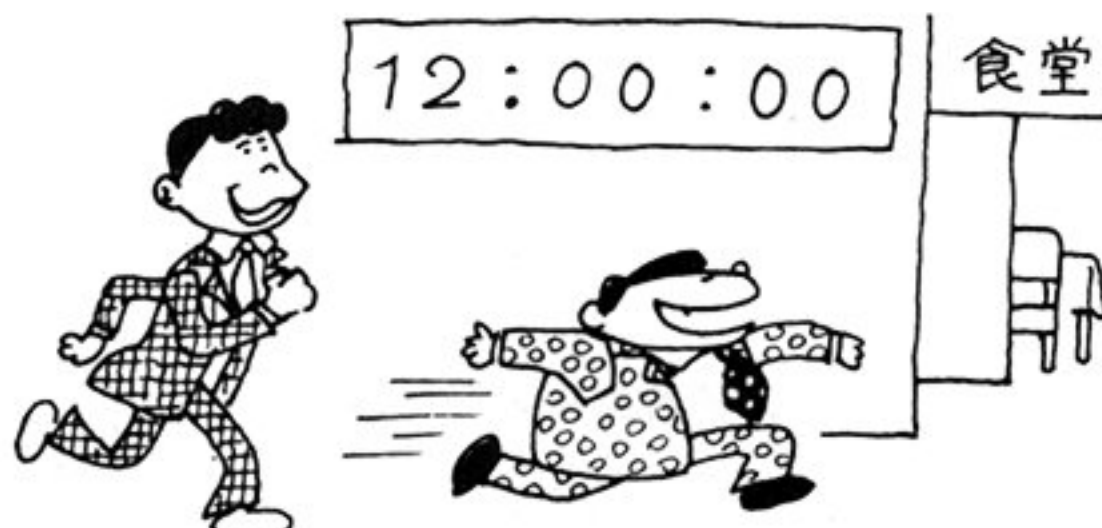
```
ON TIME GOSUB 行番号
```

行番号は，このサブルーチンの先頭に置きます．この文で定義しておくと，TIME ON文のステートメントのあとで，時計の時刻がTIME文に指定した時刻になったときに，サブルーチンの方に実行が移ります．また割り込みルーチンからの復帰は，RETURNによります．

◆ TIME

この命令文によって割り込みをする時刻を与えます．

```
TIME "hh : mm : ss"
```

hhは時刻，mmは分，ssは秒です．TIME ONのステートメントで割り込みするように設定したあと，TIME文で与えた時刻に，ON TIME GOSUBで定義したサブルーチンの実行に移ります．

◆ TIME ON

この命令文を実行したあとに，タイマ割り込みができるようになります．もちろん，割り込みサブルーチンはON TIME GOSUB 文で定義されていなくてはなりません．サブルーチンを実行中は，それまでのプログラムは，当然RETURN文でサブルーチンから復帰するまで中断されます．また復帰する場所は，RETURNのあとに行番号が書いてあればそこへ，もしなければGOSUB文の直後の文です．

◆ TIME OFF

この命令文によって，それ以前に設定されていたTIME ONを解除し，タイマ割り込みができなくなります．

◆ TIME STOP

タイマ割り込みを一時停止します．すなわち，TIME文で与えられた時刻は記憶しているが，割り込みはこのステートメントに出会うとできなくなります．割り込みが行なわれるのは，次のTIME ON文に出会ったあとです．

◆ では，実際のプログラム例を調べてみましょう．

```
10 TIME$="00:00:00"
20 CLS
30 ON TIME GOSUB 80
40 TIME"00;00;10"
50 TIME ON
60 LOCATE 10,10:PRINT TIME$
70 GOTO 60
80 LOCATE 25,10:PRINT TIME$
90 BEEP 1:FOR I=1 TO 5000
     :NEXT I
100 BEEP 0
110 RETURN
```

行番号10で，内蔵時計の時間を設定します．行番号30で，ON TIME GOSUBで割り込みがかかったときのサブルーチンへのジャンプを指定しています．そして，行番号40で割り込みを発生させる時間を指定しています．

TIMEがこの値になると，サブルーチン（行番号80～110）に移動するわけです．サブルーチンの仕事はBEEPを鳴らす仕事です．

このように，ON TIME GOSUB文を使うことにより，ふだんは普通の仕事をしていても，ある時刻になると，全く別の仕事をさせることができるわけです．ここでは，わかりやすくするために，ふだんは1秒きざみの時間を表示していて，10秒たったらピーを鳴らします．

```
10 TIME$="00:00:00"
20 CLS
30 ON TIME GOSUB 80
40 TIME"00;00;10"
50 TIME ON
60 LOCATE 15,8:PRINT TIME$
70 GOTO 60
80 SYMBOL(75,120),
     TIME$,7,4,4
90 BEEP 1:FOR I=1 TO 5000
     :NEXT I
100 BEEP 0
110 RETURN
```

```
10 TIME$="00:00:00"
20 CLS
30 ON TIME GOSUB 80
40 TIME "00:00:05"
50 TIME ON
60 LOCATE 10,10:PRINT TIME$
70 GOTO 60
80 LOCATE 10,10:PRINT TIME$
90 FOR I=1 TO 100:BEEP 1
   :NEXT I
100 TIME "00:00:09"
110 BEEP 0
120 RETURN
```

上はピーを 2回，
下はピーを 3回以上
ならすときのプログラムです．
このようにすればTIMEを
何回も設定し直す
ことができます．

```
10 TIME$="00:00:00"
20 CLS
30 ON TIME GOSUB 80
40 TIME "00:00:05"
50 TIME ON
60 LOCATE 10,10:PRINT TIME$
70 GOTO 60
80 LOCATE 10,10:PRINT TIME$
90 FOR I=1 TO 100:BEEP 1
  :NEXT
100 A=A+1
110 IF A=1 THEN
      TIME "00:00:10"
120 IF A=2 THEN
      TIME "00:00:15"
130 BEEP 0
140 RETURN
```

4.13 インターバルタイマ割り込み

一定の時間を経過するごとに，割り込みの仕事をする命令です．

ON INTERVAL GOSUB 行番号

Intervalは，間隔という意味ですね．前述のON TIME GOSUB文によく似ていますが，一定間隔ごとに割り込みの仕事をする点が違います．

INTERVAL 割り込み間隔

割り込み間隔は秒単位であり，最大65535秒までの指定が可能です．

◆ INTERVAL ON

ON TIME GOSUBを使った命令文のTIME ONの場合と同様，INTERVALで指定された時間間隔での割り込みができるようになります．

◆ INTERVAL OFF

TIME OFFの場合と同様，この命令文によってON INTERVAL GOSUBの割り込みが解除します．

◆ INTERVAL STOP

TIME STOPの場合と同様に，割り込み動作を一時的に停止するのに使われます．すなわち，コンピュータは割り込み指令は受けますが，実行はしません．割り込み再開は，次のINTERVAL ONの命令があったときです．

◆ では，実際のプログラム例を調べてみましょう．

```
10 TIME$="00:00:00"
20 COLOR7,0
30 CLS:I=1
40 ON INTERVAL GOSUB 90
50 INTERVAL 10
60 INTERVAL ON
70 LOCATE10,10,0:PRINT
   TIME$
80 GOTO 70
90 COLOR I,0
100 I=I+1
110 IF I=8 THEN I=1
120 RETURN
```

10秒ごとに表示文字の色を変えるプログラムです．別の仕事つまりサブルーチンは，行番号90～120です．

TIME $で内蔵時計の設定，行番号40で割り込みがかかったときのサブルーチンの行番号を指定します．行番号50がINTERVALの時間設定のプログラムです．タイマ割り込みの場合と非常によく似ていますね．

4.14 ファンクションキー割り込み

ファンクションキーを使って，このキーが押されたとき，定義されている別の仕事をします．

> **ON KEY（ファンクションキー番号）**
> **GOSUB　行番号**

次に示すKEY（n）ONの状態のとき，このファンクションキーを押すと，それまで続けていたプログラムを一時中断し，指定されたサブルーチンの方にジャンプします．

◆ KEY (n) ON

この命令文のあとで，[PFn]のキーを押すと，サブルーチンが割り込みます．

◆ KEY (n) OFF

ファンクションキーの割り込みができなくなります．

◆ KEY (n) STOP

ファンクションキーからの割り込みを，次にKEY (n) ONがくるまで中止します．

これらの動作は，プログラムを走らせている間だけで，その場合は通常でも，ファンクションキーの操作はできないわけです．したがってプログラムを作っている場合は，もちろん通常どおり働いてくれます．ただし，一度プログラムを走らせたとき，もしKEY (n) OFFがないと，ファンクションキーには鍵がかかったままの状態になります．

これは，KEY (n) STOPのままの状態でも同じです．必ずKEY (n) OFFを実行するようにしなくてはなりません．

◆ では例によってKEY (n) ONを使ったプログラム例を調べてみましょう．

```
10 ON KEY(1) GOSUB 120
20 ON KEY(2) GOSUB 140
30 KEY(1) ON
40 KEY(2) ON
50 FOR I=1TO 10000
60 A#=A#+I
70 NEXT
80 KEY(1) OFF
90 KEY(2) OFF
100 PRINT "1 ｶﾗ 10000 ﾏﾃﾞ
    ﾉ ｺﾞｳｹｲ"; A#
110 END
120 PRINT "ｲﾏﾏﾃﾞ ﾉ ｺﾞｳｹｲ";A#
130 RETURN
140 PRINT "ｹｲｻﾝ ｶｲｽｳ";I
150 RETURN
```

ファンクションキー1，2を割り込み用に使ったプログラムです．プログラムは1＋2＋……＋10000という長い加算です．行番号10，20でそれぞれKEY（1）とKEY（2）の割り込み用サブルーチンを指定しています．

そして，行番号120～のサブルーチンは，メインルーチンである行番号50～70の中で，今までの加算の合計はいくらになっているかを教えてくれます．また行番号140～は，FOR～NEXTが今のところ何回まわっているかを教えてくれるサブルーチンです．


```
RUN
ｲﾏﾏﾃﾞ ﾉ ｺﾞｳｹｲ 83028
ｹｲｻﾝ ｶｲｽｳ 429
ｲﾏﾏﾃﾞ ﾉ ｺﾞｳｹｲ 1258491
ｹｲｻﾝ ｶｲｽｳ 1600
ｲﾏﾏﾃﾞ ﾉ ｺﾞｳｹｲ 8592585
ｹｲｻﾝ ｶｲｽｳ 4149
ｲﾏﾏﾃﾞ ﾉ ｺﾞｳｹｲ 19291366
ｹｲｻﾝ ｶｲｽｳ 6218
ｲﾏﾏﾃﾞ ﾉ ｺﾞｳｹｲ 42869170
ｹｲｻﾝ ｶｲｽｳ 9271
1 ｶﾗ 10000 ﾏﾃﾞ ﾉ ｺﾞｳｹｲ
                    50005000
```

4.15 PRINT USING

文字または数値を，指定した書式で画面に表示します．

> **PRINT USING**
> **フォーマット文字列；文字 or 数値**

文字，数値はセミコロン；またはコンマ，で区切られます．直接モードの例によって，どのような働きをするのか，またフォーマット指定の方法を調べてみましょう．

◆ 文字列の場合

（1）　!

与えられた文字列の最初の一文字だけが表示されます．

```
A$="ABCDEFGHIJK"

Ready
PRINT USING"!";A$
A

Ready
```

（2）　&n個の空白&

n個の空白（n≧0）を″&″で囲みます．このとき与えられた文字列の先頭から，n＋2個の文字が出力されます．もし与えられた文字列がn＋2より短い場合は，左に文字を詰め，残りは空白が表示されます．

```
A$="ABCDEFGHIJK"

Ready
PRINT USING"&       &";A$
ABCDEFG
         7文字

Ready
A$="AB"

Ready
PRINT USING"&       &";A$;"12"
AB     12
  5文字
Ready
```

◆ 数値の場合

（1）　#（ナンバ記号）

ナンバ記号の個数によって表示する数字の桁数を示します．数値は右詰めで表示され，左側のあいた部分は空白が詰められます．

負の数値は負符号が先頭の数字の左に表示され，この符号も1桁として数えます．

```
10 A(1)=12
20 A(2)=12345
30 A(3)=123456
40 A(4)=1234567
50 A(5)=1.2345
60 A(6)=-123
70 A(7)=-123456
80 FOR I=1 TO 7
90 PRINT USING"#####";A(I)
100 NEXT I
RUN
   12
12345
%123456
%1234570
    1
 -123
%-123456
```


（2）　. 小数点

数値領域の任意のところに小数点をそう入することを指示します．小数点に続いてナンバ記号(#)がある場合は，小数点以下に指定された桁数分表示されます．

```
10 A(1)=123
20 A(2)=12.34
30 A(3)=123.456
40 A(4)=-0.123
50 A(5)=-0.0015
60 A(6)=-0.0001
70 FOR I=1 TO 6
80 PRINT USING
    "###.###";A(I)
90 NEXT I
RUN
123.000
 12.340
123.456
 -0.123
 -0.002
 -0.000
```

（3） ＋（プラス）

フォーマット文字列の最初または最後のプラス記号は，出力する数値の前またはあとに正負の符号を付けて表示します．

```
10 A(1)=12
20 A(2)=123
30 A(3)=123.4
40 A(4)=-0.123
50 A(5)=-123
60 A(6)=12345
70 FOR I=1 TO 6
80 PRINT USING"###+";A(I)
90 NEXT I
100 PRINT
110 FOR I=1 TO 6
120 PRINT USING"+###";A(I)
130 NEXT I
RUN
 12+
123+
123+
  0-
123-
%12345+

 +12
+123
+123
  -0
-123
%+12345
```

（4） －（マイナス）

フォーマット文字列の最後のマイナス記号は，負の数値のあとにマイナス符号を付けることを指定します．

```
10 A(1)=12
20 A(2)=12345
30 A(3)=-1234
40 A(4)=-123
50 A(5)=-123456
60 A(6)=123456
70 FOR I=1 TO 6
80 PRINT USING"#####-";A(I)
90 NEXT I
RUN
   12
12345
 1234-
  123-
%123456-
%123456
```

（5） ＊＊（2個のアスタリスク）

フォーマット文字列の最初に置かれ，数値領域の上位桁の空白部分にアスタリスクが表示されます．

2個のアスタリスクは，2桁分の領域を確保します．

```
10 A(1)=12
20 A(2)=12345
30 A(3)=12.34
40 A(4)=-123
50 A(5)=-123456
60 A(6)=1234567
70 FOR I=1 TO 6
80 PRINT USING"*******";A(I)
90 NEXT I
RUN
*****12
**12345
*****12
***-123
-123456
1234570
```

（6） ￥￥（2個の円記号）

フォーマット文字列の最初に二つの円記号を書くと，出力される数値の上位桁の直前に円記号を表示します．

二つの円記号で2桁分の領域を確保しますが，￥記号も1桁分となります．

```
10 A(1)=12
20 A(2)=12345
30 A(3)=12.34
40 A(4)=-123
50 A(5)=-123456
60 A(6)=1234567
70 FOR I=1 TO 6
80 PRINT USING"¥¥#####";A(I)
90 NEXT I
RUN
    ¥12
 ¥12345
    ¥12
  -¥123
%-¥123456
%¥1234570
```

（7）　＊＊￥（2個のアスタリスクと円記号）

フォーマット文字列の最初に＊＊￥を書くと，(5)，(6)の両方の機能をはたします．＊＊￥で3桁分の領域を確保しますが，￥記号も1桁分取ります．

```
10 A(1)=12
20 A(2)=12345
30 A(3)=12.34
40 A(4)=-123
50 A(5)=-123456
60 A(6)=1234567
70 FOR I=1 TO 6
80 PRINT USING"**¥####";A(I)
90 NEXT I
RUN
****¥12
*¥12345
****¥12
**-¥123
%-¥123456
%¥1234570
```

（8）　,（コンマ）

小数点位置指定の“.”の左側に置かれ，整数部を右側から3桁ごとにコンマで区切ります．コンマを表示するたびにその桁が確保されます．

```
10 A(1)=12345600
20 A(2)=12345.6
30 A(3)=1.23456
40 A(4)=0.123456
50 A(5)=-123456
60 A(6)=-1.23456
70 FOR I=1 TO 6
80 PRINT USING"#######,.##";
   A(I)
90 NEXT I
100 PRINT
110 FOR I=1 TO 6
120 PRINT USING"##########,";
    A(I)
130 NEXT I
RUN
%12,345,600.00
  12,345.60
        1.23
        0.12
-123,456.00
      -1.23
                12,345,600
                    12,346
                         1
                         0
                  -123,456
                        -1
```

（9）　∧∧∧∧（4個の矢印）

ナンバ記号（#）のあとに置かれます．この指定により，指数形式で表示できます．4個の矢印は，E±nnを表示する領域を確保します．ナンバ記号により，有効数字が指定され，指数はそれに合わせて調整されます．

```
10 A(1)=12345600
20 A(2)=12345.6
30 A(3)=1.23456
40 A(4)=0.123456
50 A(5)=-123456
60 A(6)=-1.23456
70 FOR I=1 TO 6
80 PRINT USING"###^^^^";A(I)
90 NEXT I
RUN
 12E+06
 12E+03
 12E-01
 12E-02
-12E+04
-12E-01
```

（10）　文字列の表示

フォーマット文字列の中に，書式指定文字以外の文字を書くと，その文字を表示します．この例ではコを表示しています．

```
10 A(1)=123456
20 A(2)=12345.6
30 A(3)=1.23456
40 A(4)=0.654321
50 A(5)=-12345
60 A(6)=-1234
70 FOR I=1 TO 6
80 PRINT USING"#####コ";A(I)
90 NEXT I
RUN
%123456コ
12346コ
    1コ
    1コ
%-12345コ
-1234コ
```

注）フォーマットで指定した桁数で出力できない場合は，数値の先頭にパーセント（%）が出力されます．%はオーバーフローを表わします．

4.16 ERROR, ERR/ERL ON ERROR GOTO

コンピュータは，あなたの命令に間違いがあると，そのむねを表示し，エラーの内容を表示してくれます．このときの表示内容とその意味は，**付録**に示してあります．

BASICでは，このようにいくつかの決められた内容を表示させること以外に，あなた自身が決めたエラーを表示させることもできるのです．

エラーに関係する命令には，次のようなものが使われます．

ERROR エラー番号

ON ERROR GOTO 行番号

RESUME [{NEXT｜行番号}]

ERR／ERL

とにかく，プログラムを走らせてみましょう．

```
10 ON ERROR GOTO 100
20 SSSS
30 END
100 IF ERR=02 THEN PRINT
      "ブンポウ ノ アヤマリ"
RUN
ブンポウ ノ アヤマリ

No Resume In 100
Ready
```

行番号10は，もしエラーに出会ったら，**付録**にあるようなエラーメッセージを表示せず，エラー処理ルーチンである行番号100に飛んで行け，という意味です．そして，行番号100にあるERRというのは，エラーが起こったとき，そのエラー番号が入る機能を持っています．

このプログラムを実行させると，行番号20にわけのわからぬ命令が書いてありますね．当然文法上のエラー（**付録**よりわかるように，エラーコード02のSyntax Errorになっている）です．そこで，ERR＝02の場合のIF文がありますから，ブンポウ　ノ　アヤマリ　と表示してきます．

しかもそれに続いて，No Resume In 100の表示が出ます．RESUMEというのは，エラー処理が終わったあと，どの行番号から実行を再開すればよいかを示す命令で，このあとに，

(1) 行番号を指定したときは，その行番号から実行を再開します．

(2) NEXTを指定したときは，エラーの起きた文の次から実行を再開します．

(3) 指定がないときは，エラーの起きた文から実行を再開します．

という機能を持っています．

たとえば，行番号100のあとに：RESUMEと入れてやると，エラーの起きた文から実行を再開するので，繰り返しエラー処理ルーチンに飛んで行き，ブンポウ　ノ　アヤマリ　という文字を無限に表示してくるはずです．したがって，ここではRESUMEのあとにNEXT，あるいは30と入れてやる必要があるわけです．

```
10 ON ERROR GOTO 100
20 INPUT X
30 Y=1/X
40 PRINT Y
50 GOTO 20
100 INPUT"0ｲｶﾞｲ ﾉ ｶｽﾞ ｦ ｲﾚﾖ "
     ;X:RESUME
RUN
? 2
 .5
? 0.1
 10
? 0
0ｲｶﾞｲ ﾉ ｶｽﾞ ｦ ｲﾚﾖ ? 10
 .1
```

1を0で割ることは許されていません．行番号30で，もしXに0が入ってきたら，本来ならエラーコード11 Division By Zeroが表示されてエラーになるところです．ここではこれを救うために，行番号100でエラー処理をし，再度インプットを促し，エラーが発生した行，つまり行番号30に戻しています．

```
10 ON ERROR GOTO 90
20 CLS
30 PRINT
     "1 ｶﾗ 5 ﾏﾃﾞ ﾉ ｽｳｼﾞ";
40 INPUT A
50 IF (A<1)OR(A>5)
     THEN ERROR 101
60 PRINT A
70 ERROR 102
80 END
90 IF ERR=101 THEN PRINT
     "ﾀﾞﾒﾃﾞｽ":RESUME 30
100 IF ERL=70 THEN PRINT
     "ｴﾗｰ ﾃｽﾄ":RESUME 80

1 ｶﾗ 5 ﾏﾃﾞ ﾉ ｽｳｼﾞ? 6
ﾀﾞﾒﾃﾞｽ
1 ｶﾗ 5 ﾏﾃﾞ ﾉ ｽｳｼﾞ? 9
ﾀﾞﾒﾃﾞｽ
1 ｶﾗ 5 ﾏﾃﾞ ﾉ ｽｳｼﾞ? 4
 4
ｴﾗｰ ﾃｽﾄ
```

ERRORは，故意にエラーを発生させるときに使われる命令で，エラー番号(0～255)を定義することができます．もし付録のエラーメッセージの中にあれば，これを指定できますし，なければ行番号50にあるように，あなた自身が勝手にエラー番号を作って付けてやることが可能です．ここではエラー番号を101に選んでいます．そして，行番号90でERR=101のときは，ダメデスというエラーメッセージを出すようになっています．

また，行番号70では，わざとERROR 102が定義されています．このエラー番号に対応するエラーメッセージではありません．もし何もない場合には，Unprintable Errorと表示されますが，この場合にはERL変数というのが用意されており，ERROR文の行番号が代入されるため，行番号100でエラーテストという，エラーメッセージが出るようになっています．

そして，同時にERRは102というエラー番号が代入されているはずです．行番号95に，もしERR=102なら〝XXX〟と表示するようなプログラムを作ってそう入して，試してみてください．なおERLは，エラーが起こった行番号が代入される変数です．

```
10 'ON ERROR GOTO 90
20 CLS
30 PRINT
     "1 ｶﾗ 5 ﾏﾃﾞ ﾉ ｽｳｼﾞ";
40 INPUT A
50 IF (A<1)OR(A>5)
     THEN ERROR 101
60 PRINT A
70 ERROR 102
80 END
90 IF ERR=101 THEN PRINT
     "ﾀﾞﾒﾃﾞｽ":RESUME 30
95 IF ERR=102 THEN PRINT
     "XXX"
100 IF ERL=70 THEN PRINT
     "ｴﾗｰ ﾃｽﾄ":RESUME 80

1 ｶﾗ 5 ﾏﾃﾞ ﾉ ｽｳｼﾞ? 6
ﾀﾞﾒﾃﾞｽ
1 ｶﾗ 5 ﾏﾃﾞ ﾉ ｽｳｼﾞ? A
?Redo From Start
? 5
 5
XXX
ｴﾗｰ ﾃｽﾄ
```

4.17 RANDOMIZE, RND

全くでたらめに並んだ数のことを乱数といいます．箱の中に0から9までの数を書いた札を入れておき，よくかきまぜてからその中の1枚を，中をのぞき込んだりしないで取り出します．その数が5だったとします．次にその札を中に戻し，再びよくかきまぜてから同じ動作を繰り返します．

このような動作を何度も繰り返して得た数の並びは，全く規則性のない乱数となるでしょう．しかし，このような数の並びをコンピュータを使って作り出してやることは困難なので，実用上この乱数と同じ結果を得るように，計算式を使って作り出しています．

乱数は，ゲームなどで規則性のない動き方をさせるときに欠かせないものですし，実用上の計算をするときも利用できるなど，コンピュータ処理にはぜひ欲しい道具なのです．

```
10 X=RND(1)
20 PRINT X
30 GOTO 10
RUN
 .591065
 .207991
 .550967
 .635863
 .10411
 .806334
 4.29073E-02
```

いつまでも，小数点を含む数字を表示してきますが，途中で停止させたものです．

RND(1)の(　)内は式でもよく，ここが省略されたり正の値であれば，上例と同じ系列の乱数を表示してきますし，もし負の値であれば，また別の系列の乱数を表示してきます．さらに0であれば，プログラムの中で一つ前に発生した乱数と同じ乱数を出します．

```
10 X=RND(1)
20 Y=RND(0)
30 PRINT X,Y
RUN
 .591065           .591065
```

```
10 FOR I=1 TO 7
20 A(I)=RND(I)
30 PRINT A(I)
40 NEXT I
RUN
 .591065
 .207991
 .550967
 .635863
 .10411
 .806334
 4.29073E-02
```

上のRUN結果と前述のRND(1)の結果とを比べてみてください．

下にいろいろな場合のRUN結果を示します．

```
10 X=RND(2)
20 Y=RND(10)
30 PRINT X,Y
40 GOTO 10
RUN
 .591065           .207991
 .550967           .635863
 .10411            .806334
 4.29073E-02       .953862
 .354289           .556498
```

```
10 X=RND(-2)
20 Y=RND(-10)
30 PRINT X,Y
40 GOTO 10
RUN
 .522223           .147223
 .522223           .147223
 .522223           .147223
```

```
10 X=RND(-3)
20 Y=RND(-20)
30 PRINT X,Y
40 GOTO 10
RUN
 .772223           .147223
 .772223           .147223
 .772223           .147223
```

表示してくる乱数が，毎回同じものであったら，乱数が乱数になりませんね．でも，安心してください．乱数の系列を変更する命令もちゃんと用意されています．これがRANDOMIZE［式］です．

式を省略すると，プログラムの実行が停止し，次の表示が行なわれます．

　　RND　Seed（−32768−32767）？

つまり式の中は，−32768〜32767の範囲でなくてはならないわけです．そして，(　)内の数値が変わるたびに違った乱数の系列が選ばれることになります．

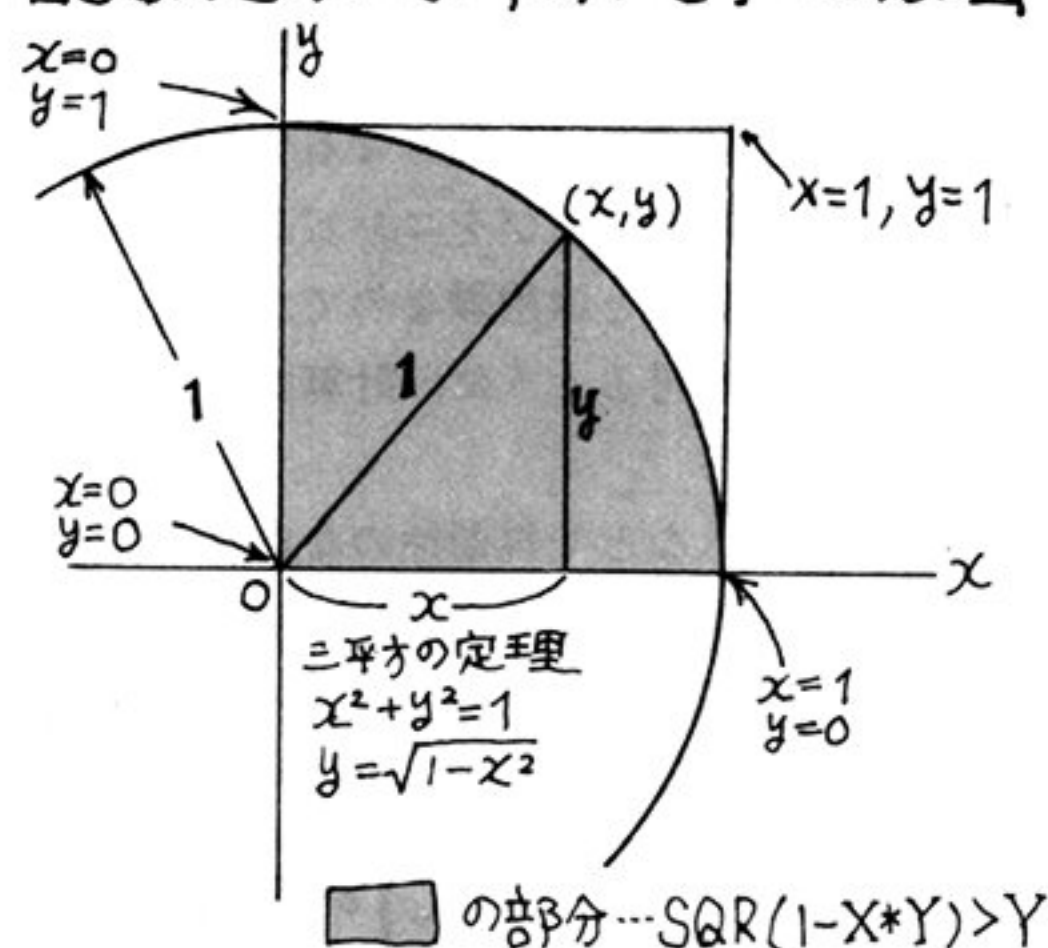
```
10 RANDOMIZE(5)
20 X=RND(1)
30 PRINT X
40 GOTO 10
RUN
 .586623
 .803328
 .591215
 .780675
```

```
10 RANDOMIZE(-5)
20 X=RND(1)
30 PRINT X
40 GOTO 10

Ready
RUN
 8.81791E-02
 .304885
 9.66781E-02
 .278325
```

また，RUNさせるたびに違った乱数系列を得たいときの一つとして，コンピュータに内蔵している時計を利用する方法もありますし，コンピュータを扱う人の名前をインプットさせ，そのアスキーコードを使ったり，生年月日を掛けたり，あるいは生年月日をアスキーコードで割った剰余を使ったりする方法もあり，そこはあなたの腕次第です．

ここでは，TIMEを使う簡単な方法を示しておきます．TIMEは1日24時間ですから，0〜86399の値になります(24×3600−1)．これでは式の値をオーバーしています．そこで，TIME＄のうち右から2桁取り出し，これを式の値に使おうというわけです．これにより，RUNさせるたびに60種の異なる乱数系列を得ることができます．

```
10 RANDOMIZE(VAL(RIGHT$
     (TIME$,2)))
20 X=RND(1)
30 PRINT X
40 GOTO 10
RUN                   RUN
 .58055                .390059
 .797255               5.27028E-02
 .88983                .442152
 .822102               .33083
 .578837               .38444
```

乱数を使って円周率を求める方法

```
10 N=0
20 INPUT I
30 RANDOMIZE(I)
40 FOR J=1 TO I
50 X=RND(1):Y=RND(1)
60 IF SQR(1-X*X)>Y THEN
     N=N+1
70 NEXT J
80 S=4*N/I
90 PRINT S
100 GOTO 10
```

行番号

10…GOTO 10で始めにもどったとき Nを0にする．

20…乱数を何回発生させるか？

50…xが1〜0，yが1〜0の範囲で乱数を発生させる．

60…その点が円の内部に入っていれば Nに1を加える．

80…4倍すると半径1の円の面積つまり円周率になる．

90…円周率の表示

100…はじめにもどれ！

4.18 CLEAR

全ての数値変数を0に，また全ての文字変数を空文字列に初期設定します．

CLEAR ［文字領域の大きさ］［，BASICの使用する上限のメモリアドレス］

全ての変数をクリアする命令であり，BASICの使用する文字領域，BASICの使用エリアを設定できます．

文字領域とは，BASICプログラムを実行するとき，文字列を取り扱う場合のワークエリアのことで，初期の段階では300に設定されています．

4.8 ストリング変数で，ジュゲムの文字の長さを出したとき，CLEAR 400としたのは，文字領域がたりなくなったため，文字領域を広げてやったわけですね．ただし，CLEAR文はプログラムの初めに入れておかないと，変数の中身までも全て0になってしまいます．

もし，プログラム実行中に，Out of String Spaceというエラー表示が起きたら，このようにして文字領域を広げてやればよいのです．

また，BASICと機械語類とをつなげて使用する場合，BASICで使用するメモリの領域を小さくしておき，メモリの空領域に機械語のサブルーチンを書き込んで使用することがあります．このときにCLEAR文を使って，BASICの使用領域を指定します．

◈ FRE

BASICプログラムで使用できるメモリのバイト数を示します．

FRE（整　数）
FRE（文字列）

大きなプログラムになると，メモリがだんだん少なくなってきて，あとどれくらいのメモリが残っているか気になってきます．そこで残りのメモリが何バイトであるかを調べるのがこの命令です．

FRE（整数）で，BASICで使用できるメモリのバイト数を示します．

FRE（文字列）で文字領域の残りのバイト数を示します．

これらのコマンドは，ほとんど直接モードで使用されます．なお，ここでいう整数は，整数であれば何でもよく，文字列も同様に″ ″でくくった文字を入れればよいのです．ここでは整数には1，文字列には″A″を使っています．

```
PRINT FRE("A")
 300
```

文字列エリアの残りを示しています．

```
PRINT FRE(1)
 30502
```

BASICのプログラムが，まだ30502バイト分入る余地があることを示しています．ただし，この例の数字はメモリの使用状態により変わってきます．

```
CLEAR 400,&H4000

Ready
PRINT FRE(1)
 14019
```

ここでは文字領域の大きさと共に，BASICの使用できる上限のメモリアドレスを指定しました．そこで，BASICプログラムの入る領域がぐんと減っていることに注意してください．

次に文字領域を調べてみましょう．上のコマンドによって文字領域は広くなっているはずです．

```
PRINT FRE("A")
 400
```

CLEAR 400，& H4000により，機械語のサブルーチンを4000以上のアドレスに書き込むことができます．なお上限のアドレスは，BASICのインタプリタが書き込まれているアドレスまでです．

4.19 CSRLIN, POS, POINT

◈ CSRLIN

画面上のカーソルの垂直位置を与えます.

```
CSRLIN
```

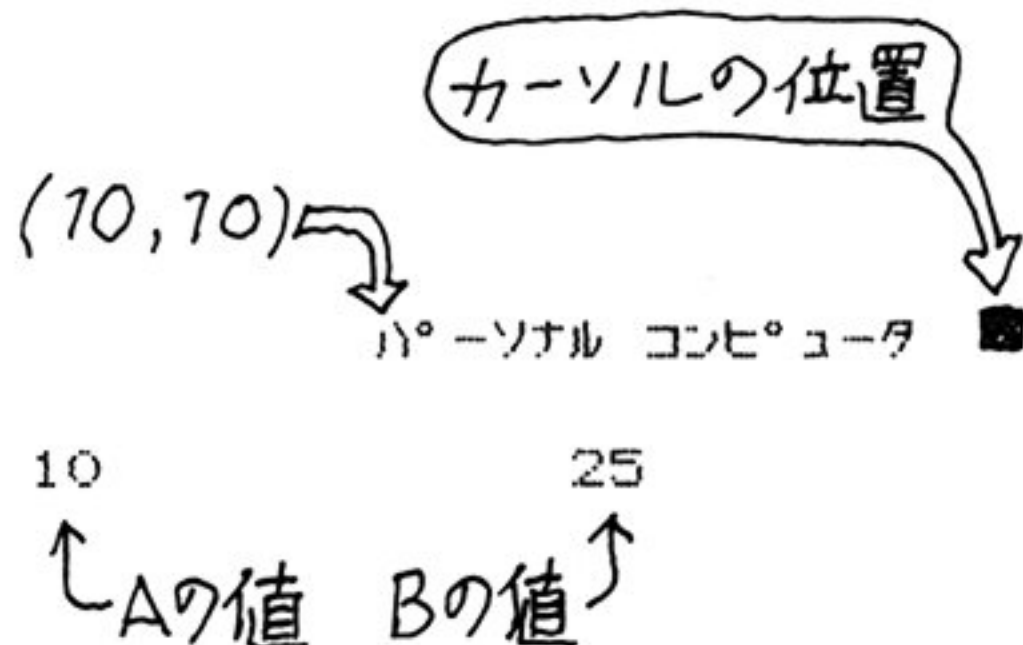
画面上のカーソルの垂直位置を行単位で返してきます.

◈ POS

画面上のカーソルの水平位置を与えます. また後述のように, ファイル番号がプリンタに割り当てられているときは, プリンタのヘッドの水平位置を与えます.

```
POS (ファイル番号)
```

ファイル番号が0のときは, 画面上のカーソルの水平位置を与えます.

```
10 CLS
20 LOCATE 10,10
30 PRINT "ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ ";
40 A=CSRLIN
50 B=POS(0)
60 PRINT
70 PRINT
80 PRINT A,B
```

このプログラム例では, 画面の10行目にパーソナル␣コンピュータ␣と表示します. そのときカーソルはコンピュータ␣のすぐあとの左から25文字目にあります. その垂直, 水平の位置をA, Bに代入して表示します.このように,カーソルがどの位置にいるかを見るときに利用するわけです.

◈ POINT

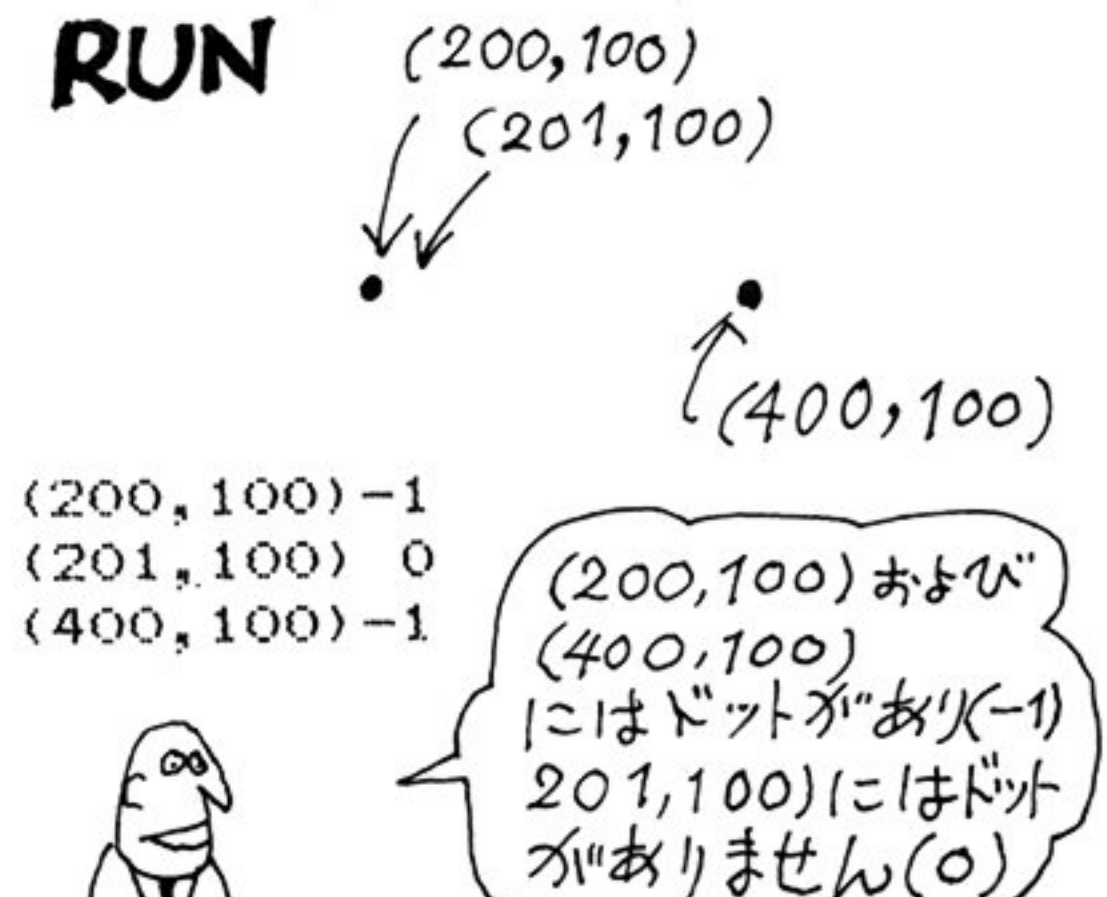
指定した位置にドットが表示しているか, いないかを与えます.

```
POINT (水平位置, 垂直位置)
```

指定した位置にドットが表示していればー1, 表示していなければ0を与えます. また背景色は0となります.

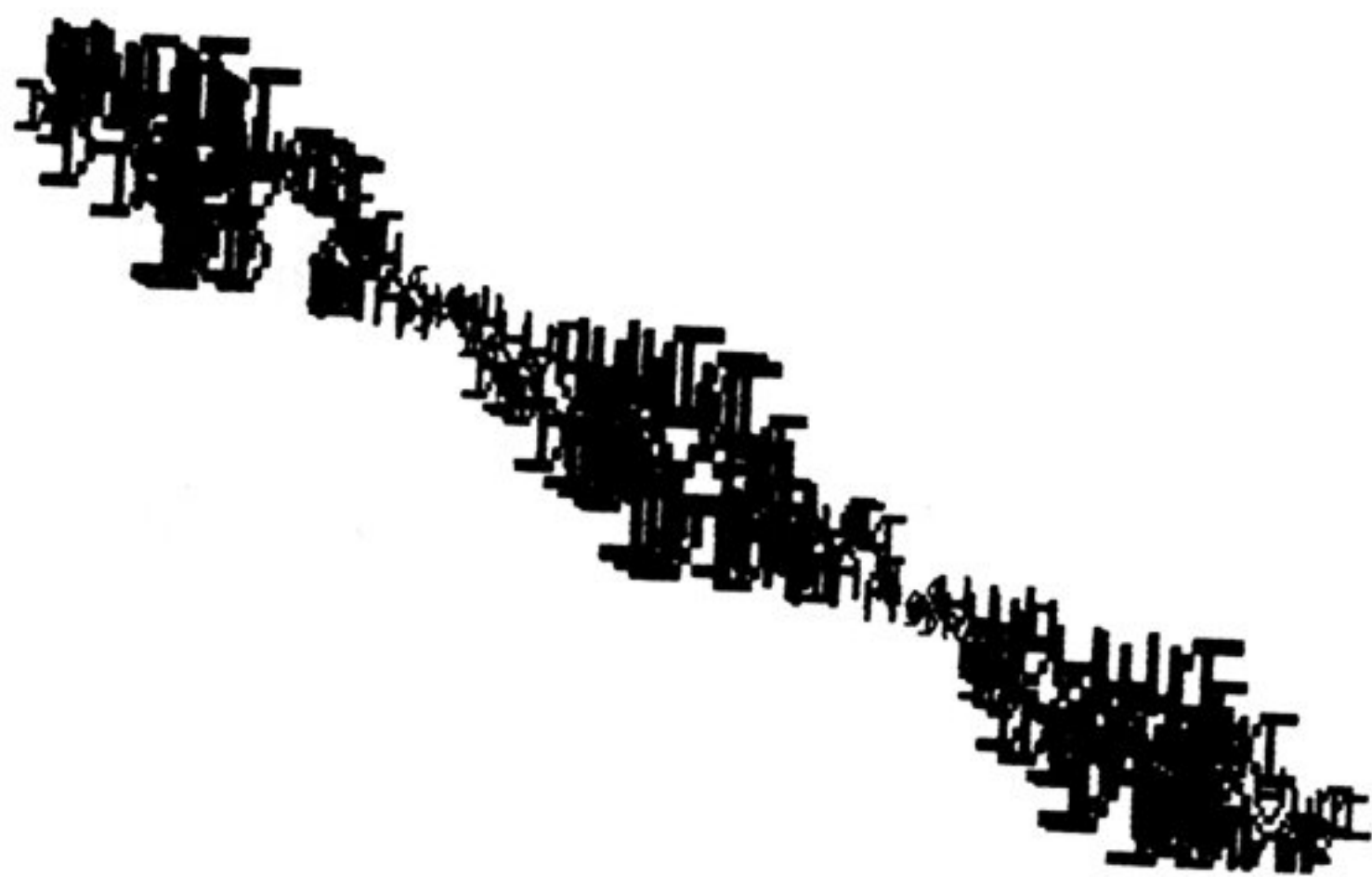
```
10 COLOR 7,0
20 CLS
30 PSET (200,100)
40 PSET (400,100)
50 A=POINT(200,100)
60 B=POINT(201,100)
70 C=POINT(400,100)
80 LOCATE 0,13
90 PRINT "(200,100)";A
100 PRINT "(201,100)";B
110 PRINT "(400,100)";C
```

```
(200,100)-1
(201,100) 0
(400,100)-1
```

(200,100)および(400,100)にはドットがあり(-1) 201,100)にはドットがありません(0)

4.20 プログラム例

```
10 COLOR 2,7
20 CLS
30 H=1:V=1
40 FOR I=0 TO 180
50 IF H>4 OR CF=1 THEN H=H-.1:CF=1ELSE H=H+.1
60 IF V>4 OR CF=1 THEN V=V-.1:CF=1ELSE V=V+.1
70 P=INT(RND(1)*4)
80 SYMBOL(X+3*I,Y+I),"A",H,V,2,P,PSET
90 IF H<1 THEN CF=0
100 IF V<1 THEN CF=0
110 SYMBOL(X+3*I,Y+I),"A",H,V,2,P,PRESET
120 NEXT
```

Aという文字が 乱数(P)によって ぐるぐる回転しながら、しかも大きく なったり 小さくなったりします。

表示されたAを消さずにそのままにしておき、ハードコピーをとった例

Aの軌跡

H…水平方向の倍率

V…垂直方向の倍率

CFが0なら増大

CFが1なら縮少

CFは文字を大きくしていくか、小さくするかを決める変数だ。

```
0 REM BIT ﾊﾟﾀｰﾝ
10 CLS
20 PRINT "0 ｶﾗ 255 ﾏﾃﾞ ﾉ ｽｳ "
30 INPUT A
40 IF A<0 OR A>255 THEN 20
50 IF 0=<A-128 THEN A=A-128:B$=B$+"  1" ELSE B$=B$+"  0"
60 IF 0=<A-64  THEN A=A-64 :B$=B$+"  1" ELSE B$=B$+"  0"
70 IF 0=<A-32  THEN A=A-32 :B$=B$+"  1" ELSE B$=B$+"  0"
80 IF 0=<A-16  THEN A=A-16 :B$=B$+"  1" ELSE B$=B$+"  0"
90 IF 0=<A-8   THEN A=A-8  :B$=B$+"  1" ELSE B$=B$+"  0"
100 IF 0=<A-4   THEN A=A-4  :B$=B$+"  1" ELSE B$=B$+"  0"
110 IF 0=<A-2   THEN A=A-2  :B$=B$+"  1" ELSE B$=B$+"  0"
120 IF 0=<A-1   THEN A=A-1  :B$=B$+"  1" ELSE B$=B$+"  0"
130 PRINT
140 PRINT "DATA BIT ﾊﾟﾀｰﾝ"
150 PRINT
160 PRINT " D7 D6 D5 D4 D3 D2 D1 D0"
170 PRINT B$
180 END
```

```
0 ｶﾗ 255 ﾏﾃﾞ ﾉ ｽｳ
? 255

DATA BIT ﾊﾟﾀｰﾝ

 D7 D6 D5 D4 D3 D2 D1 D0
  1  1  1  1  1  1  1  1

0 ｶﾗ 255 ﾏﾃﾞ ﾉ ｽｳ
? 82

DATA BIT ﾊﾟﾀｰﾝ

 D7 D6 D5 D4 D3 D2 D1 D0
  0  1  0  1  0  0  1  0
```

0から255までの数を，8ビットの2進数（D_7, D_6, ……D_1, D_0）に直すプログラムです．
INPUTされた数からつぎつぎに
128, 64, 32, 16, 8, 4, 2, 1
を引いてみます．そして引けたら1を引けなかったら0を立てるのです．

同じ仕事を下のプログラムでもやってくれるぞ．

```
0 REM BIT ﾊﾟﾀｰﾝ
10 CLS
20 PRINT "0 ｶﾗ 255 ﾏﾃﾞ ﾉ ｽｳ "
30 INPUT A
40 IF A<0 OR A>255 THEN 20
50 FOR I=7 TO 0 STEP-1
60 IF 0=<A-2^I THEN A=A-2^I:B$=B$+"  1" ELSE B$=B$+"  0"
70 NEXT
80 PRINT
90 PRINT "DATA BIT ﾊﾟﾀｰﾝ"
100 PRINT
110 PRINT " D7 D6 D5 D4 D3 D2 D1 D0"
120 PRINT B$
130 END
```

```
10 CLS
20 LOCATE 2,7
30 PRINT "ｱﾅﾀ ﾉ ｼｺﾞﾄｼﾞｶﾝ ﾊ ";
40 INPUT I1
50 LOCATE 2,9
60 PRINT "ｱﾅﾀ ﾉ ﾂｳｷﾝ ｼﾞｶﾝ ﾊ ";
70 INPUT I2
80 LOCATE 2,11
90 PRINT "ｱﾅﾀ ﾉ ｽｲﾐﾝｼﾞｶﾝ ﾊ ";
100 INPUT I3
110 IF (I1+I2+I3)>24  THEN10
120 CLS:COLOR 4,0
130 PRINT@(240,0),&H2422,&H2120,&H244A,&
H2120,&H243F,&H2120,&H244E,&H2120,&H2332
,&H2120,&H2334,&H2120,&H3B7E,&H2120,&H34
56
140 COLOR 10,0
150 PRINT@(8,120),&H2120,&H2120
160 COLOR 2
170 PRINT@(48,120),&H3B45,&H2120,&H3B76
180 COLOR14,0
190 PRINT@(8,138),&H2120,&H2120
200 COLOR 6,0
210 PRINT@(48,138),&H444C,&H2120,&H3650
220 COLOR 12,0
230 PRINT@(8,156),&H2120,&H2120
240 COLOR 4,0
250 PRINT@(48,156),&H3F67,&H2120,&H4C32
260 COLOR 13,0
270 PRINT@(8,174),&H2120,&H2120
280 COLOR 5,0
290 PRINT@(48,174),&H243D,&H244E,&H423E
300 A0=0
310 A1=I1/24
320 A2=A1+I2/24
330 A3=A2+I3/24
340 CIRCLE (380,110),190,2,.4495,A0,A1
     ,F
350 CIRCLE (380,110),190,6,.4495,A1,A2
     ,F,PSET
360 CIRCLE (380,110),190,4,.4495,A2,A3
     ,F,PSET
370 CIRCLE (380,110),190,5,.4495,A3,1,F
     ,PSET
380 COLOR 7,0
390 END
```

RUNによりあなたの一日の仕事時間、通勤時間、睡眠時間をきいてきます。そこで時間数を入れてやると、一日24時間の使われ方の比率を、きれいなカラー円形グラフで表示してくれます。表示文字は漢字まじりの日本語です。

&H2120はスペース. たとえば140 COLOR 10,0のようにベースの色を指定しているので&H2120の部分に色が出ます.

もっとこまかく分類することも簡単にできます。やってみてください。

あなたの24時間

```
0 REM スウヒョウX,X*X,X*X*X,SQR(X)
10 CLS
20 PRINT "  X  X*X X*X*X  SQR(X)"
30 FOR I=1 TO 20
40 PRINT USING "###";I;
50 PRINT USING "  ###";I*I;
60 PRINT USING "  ####";I*I*I;
70 PRINT USING "  #.###";SQR(I)
80 NEXT
90 GOTO 90
```

X	X*X	X*X*X	SQR(X)
1	1	1	1.000
2	4	8	1.414
3	9	27	1.732
4	16	64	2.000
5	25	125	2.236
6	36	216	2.449
7	49	343	2.646
8	64	512	2.828
9	81	729	3.000
10	100	1000	3.162
11	121	1331	3.317
12	144	1728	3.464
13	169	2197	3.606
14	196	2744	3.742
15	225	3375	3.873
16	256	4096	4.000
17	289	4913	4.123
18	324	5832	4.243
19	361	6859	4.359
20	400	8000	4.472

PRINT USINGを使ってフォーマットをきれいに揃えるプログラムの例です.

```
0 REM スウ ノ ヘイキン ゴウケイ
10 CLS
20 N=9999:M=-9999
30 PRINT "4ケタマデ ノ スウジ ヲ イレテクダサイ ";
40 INPUT A
50 IF A=9999THEN 110
60 C=C+1
70 T=T+A
80 IF M<A THEN M=A
90 IF N>A THEN N=A
100 GOTO 30
110 PRINT
120 PRINT "ニュウリョクスウ";C
130 PRINT " TOTAL=";T
140 PRINT "   MAX=";M
150 PRINT "   MIN=";N
160 PRINT "  ヘイキン=";T/C
```

```
4ケタマデ ノ スウジ ヲ イレテクダサイ ? 12
4ケタマデ ノ スウジ ヲ イレテクダサイ ? 55
4ケタマデ ノ スウジ ヲ イレテクダサイ ? -56
4ケタマデ ノ スウジ ヲ イレテクダサイ ? 8
4ケタマデ ノ スウジ ヲ イレテクダサイ ? 0.123
4ケタマデ ノ スウジ ヲ イレテクダサイ ? 6
4ケタマデ ノ スウジ ヲ イレテクダサイ ? 9999

ニュウリョクスウ 6
 TOTAL= 25.123
   MAX= 55
   MIN=-56
  ヘイキン= 4.18717
```

4けたの数値(-9999をこえ9999未満)をINPUTし,合計,最大,最小,平均値を求めるプログラムです.9999をINPUTすると行番号110に進み結果をCRT上に表示してくれます.

```
0 REM ﾃﾞﾝﾀｸ
10 CLS
20 A$=INKEY$
30 IF A$="" THEN 20
40 IF (ASC(A$)<42) OR (ASC(A$)>57)XOR(A$="=") THEN 20
50 PRINT A$;
60 IF A$="+" THEN GOSUB 130:CC=0:GOTO 20
70 IF A$="-" THEN GOSUB 130:CC=1:GOTO 20
80 IF A$="*" THEN GOSUB 130:CC=2:GOTO 20
90 IF A$="/" THEN GOSUB 130:CC=3:GOTO 20
100 IF A$="=" THEN GOSUB130:GOTO 190
110 C$=C$+A$
120 GOTO 20
130 IF CC=0 THEN B$=STR$(VAL(B$)+VAL(C$))
140 IF CC=1 THEN B$=STR$(VAL(B$)-VAL(C$))
150 IF CC=2 THEN B$=STR$(VAL(B$)*VAL(C$))
160 IF CC=3 THEN B$=STR$(VAL(B$)/VAL(C$))
170 C$=""
180 RETURN
190 PRINT B$
200 C$="":B$="":CC=0
210 PRINT
220 GOTO 20
```

```
1+2+3+4+5+6+7+8+9+10= 55

1*5= 5

5*55= 275

10+100*200= 22000

10/3= 3.333333333333333

8+4-5*10/5= 14
```

コンピュータを電卓の操作と同じようなやり方で計算させようという変なプログラムです．キーからの入力はINKEY$により文字変数として扱い，＋，－，＊，／がキーインされるたびに計算します．結果は＝キーのキーインにより表示しています．行番号40は，関係のないキーインがあったとき，無視するためのものです．

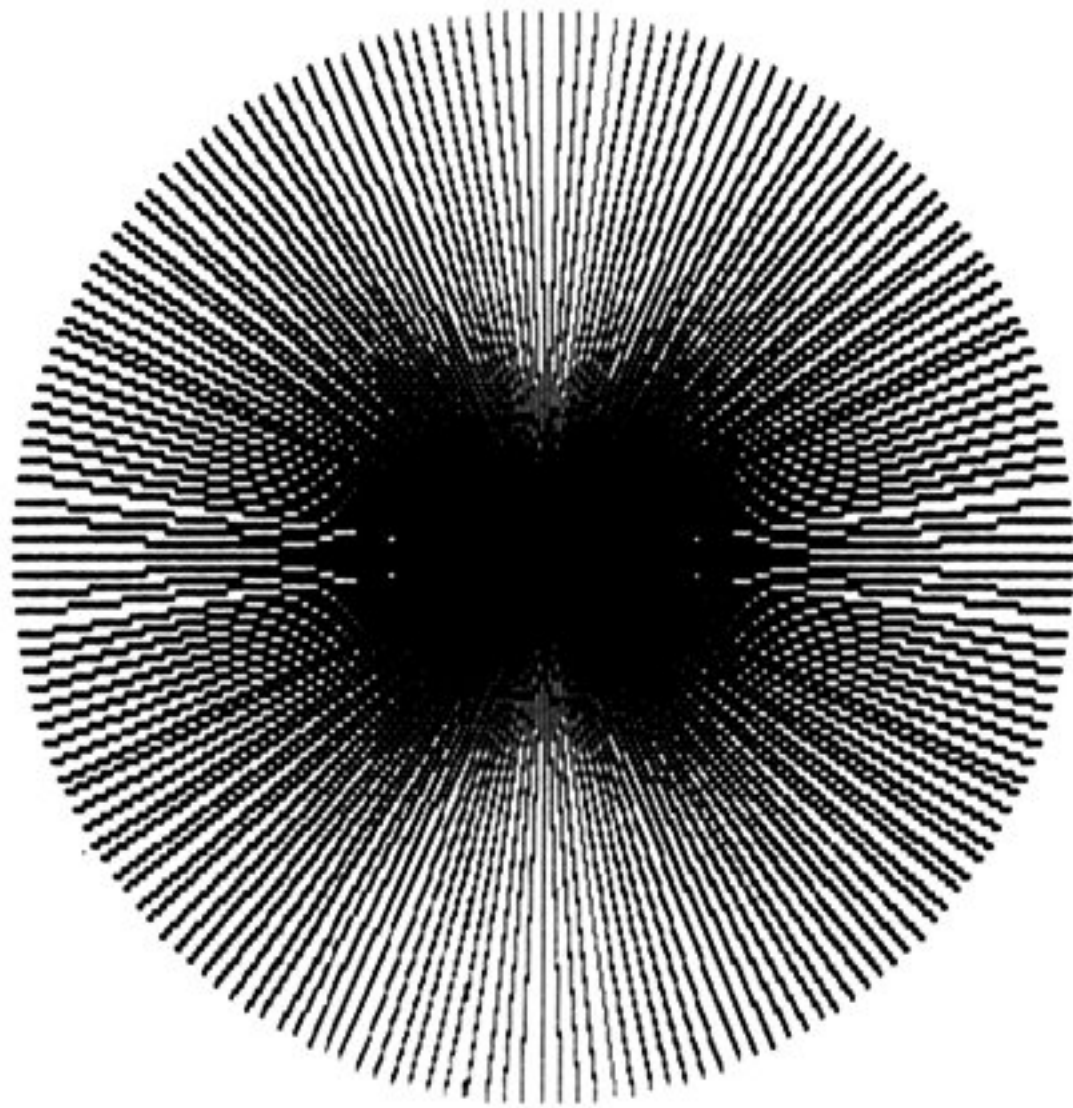
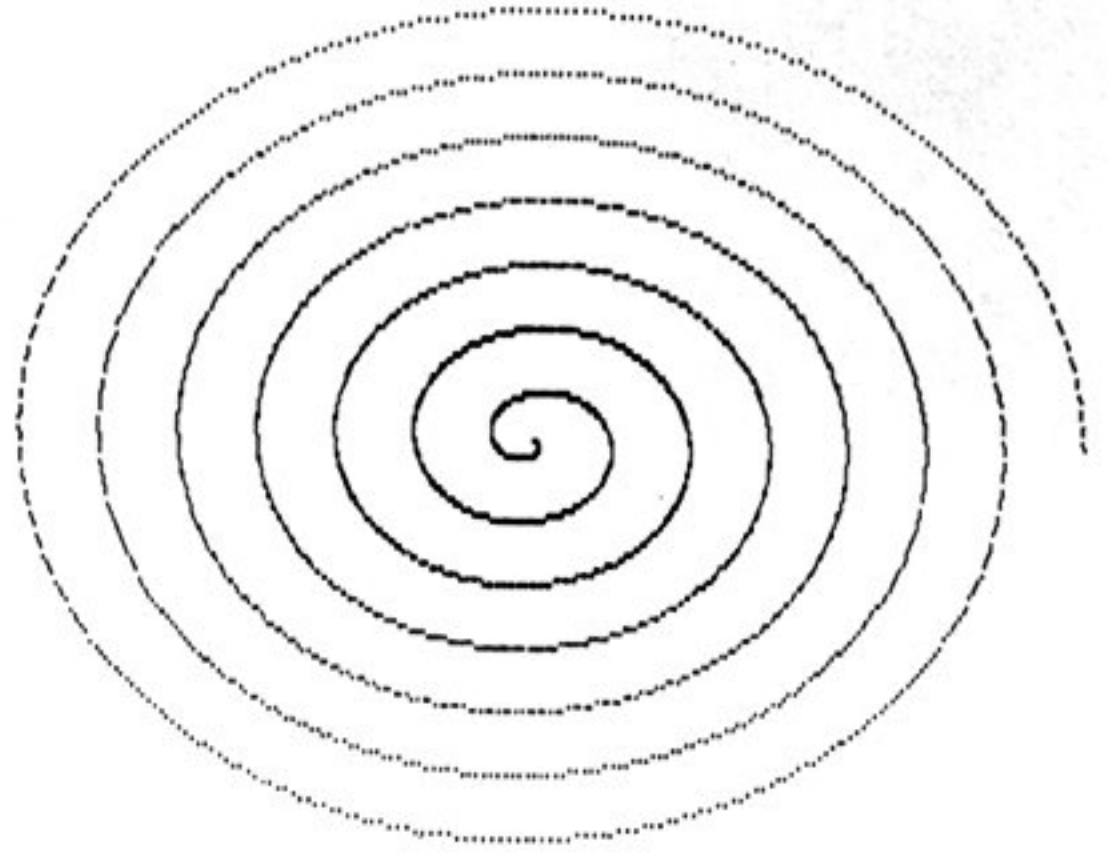
電卓的な使い方のため上式はこれで正しいのです．たとえば
10+100*200= 22000
は，本当は
(10+100)*200= 22000
となるのです．

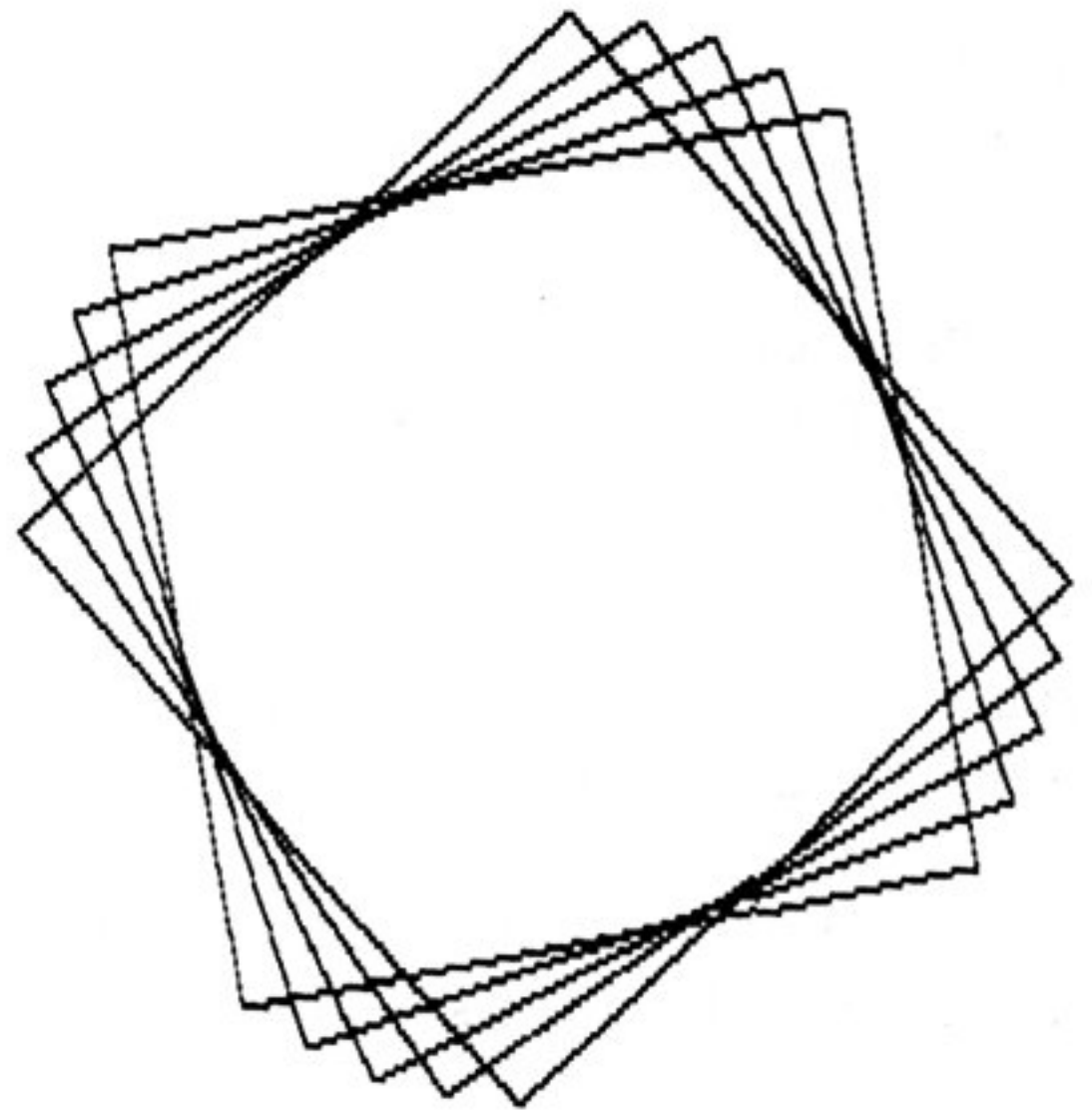
```
0 REM ウズマキ
10 CLS
20 FOR C=0 TO 6
30 FOR R=0 TO 360
40 X=320+(C*36+R/10)*COS(R*3.14159/180)
50 Y=99+(C*12+R/30)*SIN(R*3.14159/180)
60 PSET (X,Y,7)
70 NEXT R,C
```

```
0 REM チョクセン ノ カイテン
10 CLS
20 A=90
30 FOR C=1 TO 7
40 FOR R=0 TO 360 STEP 2
50 X=320+210*COS((I*A+R)*3.14159/180)
60 Y=90+90*SIN((I*A+R)*3.14159/180)
70 LINE@(320,90)-(X,Y),PSET,C
80 NEXT R,C
```

```
0 REM セイホウケイ ノ カイテン
10 CLS
20 A=90
30 FOR C=1 TO 7
40 FOR R=0 TO 360 STEP 8
50 FOR I=0 TO 3
60 X(I)=320+210*COS((I*A+R)*3.14159/180)
70 Y(I)=90+90*SIN((I*A+R)*3.14159/180)
80 NEXT I
90 CONNECT (X(0),Y(0))-(X(1),Y(1))-(X(2)
   ,Y(2))-(X(3),Y(3))-(X(0),Y(0)),C,PSET
100 NEXT R,C
```

sin cos を使うと円形の図形を描くことができます.上の三つのプログラムはその中の一例です.実際にインプットして試してください.
このようなパターン図形の場合忘れてならないのはX軸,Y軸の長さが異なる点です.

ウズマキのプログラムの例です。1°ステップごとにPSETによってプロットしていきます。ステップ数をもっと小さくすると細かい曲線に、ステップ数を大きくすると粗い曲線になります。

チョクセンノ カイテンの例です。放射状の直線が次第に色を変えていく美しいパターンです。

セイホウケイノ カイテン。たくさんの正方形が次第に色を変えていきます。図は途中の段階でSTOPさせハードコピーしたものです。

5 入出力装置とのやり取り

各種の入出力装置とのやり取りについて，その基本的な命令を述べてあります．入出力装置からの読み出しや書き込みは，私たちが普通のファイルを開いて読んだり，書き込んだりする操作とよく似ています．この入門書で，ファイルの全てを説明しつくすことは困難なため，ここに示した基本的な考え方を理解していただき，より高度な利用への基礎にしていただきたいと思います．

```
DISK VERSION
How many disk drives       ?
How many disk files(0-15)?

FUJITSU F-BASIC Version 3.0
Copyright (C) 1981 By FUJITSU MICROSOFT
25775 Bytes Free

Ready
```

フロッピィディスクを利用する場合の CRT 表示
（DISK モード）

How many disk drives (1−4) ?
と表示が出る．そこでドライブ数を入れ RETURN を押す．もう 1度 RETURN すると DISK モードになる．

5.1 OPEN, CLOSE

FMシリーズでは，全ての入出力装置をファイルという概念で扱っていることは，すでに述べたとおりです．ひと口に入出力装置といっても，いろいろな種類のものがありますが，これらのファイルに対して，プログラムの処理結果を書き込んだり，あるいは読み出したりするときは，ファイル番号を決め，これによって実行されます。

ファイル番号というのは，ファイルディスクリプタに対して割り当てられた番号で，次の OPEN 文によって定義されるわけです．

なお，ファイルディスクリプタについては，**2.14** を参照してください．

OPEN″モード″，［#］ファイル番号，″ファイルディスクリプタ″

OPEN は，ファイルのオープン処理を実行します．すなわちファイルディスクリプタで示される機器とのデータの入出力方法を定義します．OPEN 文で定義したあとは，ファイル番号で呼び出し，入出力を可能にします．

つまり，INPUT # PRINT #, LINE INPUT #, PRINT USING #, GET, PUT により，ファイルの入出力を実行する場合は，その前にまず OPEN 文を実行しなくてはならないのです．

モードは，ファイルにインプットするのか，アウトプットするのかを指定します．

I（インプット）：シーケンシャルファイルからのインプットを実行します．

O（アウトプット）：シーケンシャルファイルへのアウトプットを実行します．

A（アペンド）：シーケンシャルファイルにデータの追加をします．

R（ランダム）：ランダムファイルへの入出力を実行します．

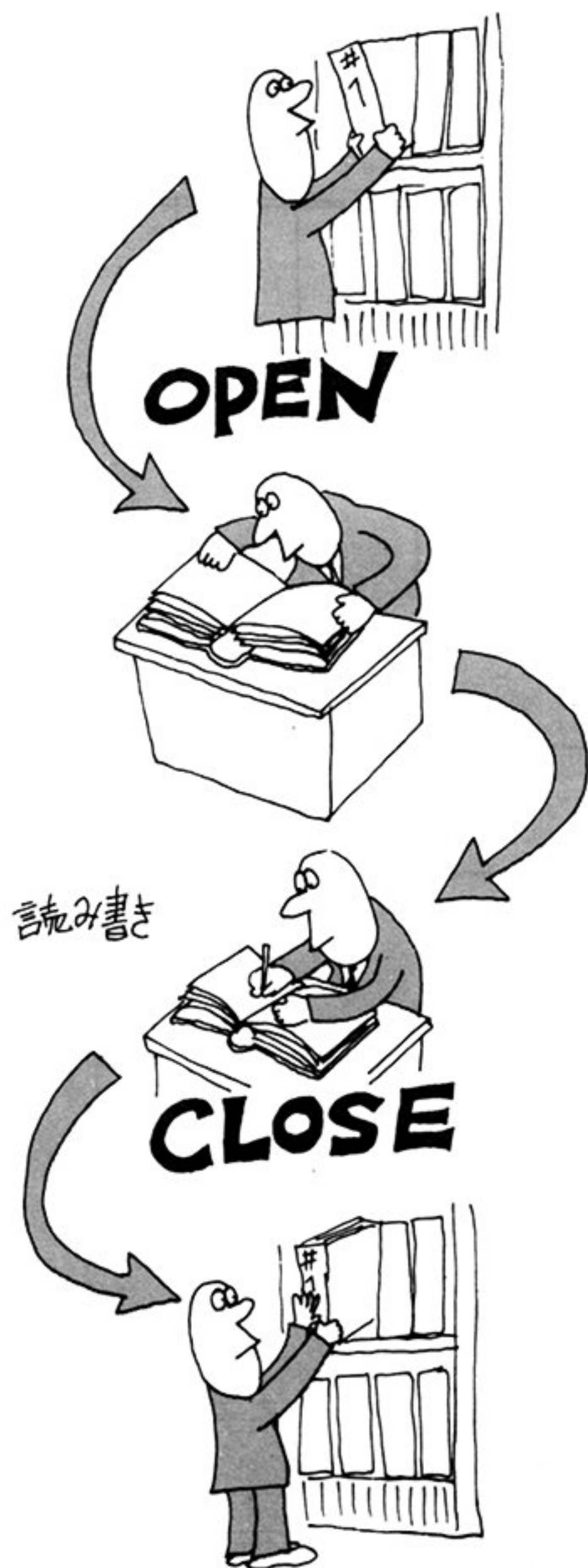
なおファイル番号は，1から 16 まで指定できます．

```
OPEN "O",#1,"LPTO:"
PRINT #1,"ABCDEFG"
```

プリンタに文字を印字する場合の OPEN 文です．＃1はプリンタに対するアウトプットと定義した上で，PRINT ＃1で印刷し出すようにしています．なお PRINT ＃は次の章を参照してください．

ファイルに対して読み書きが完了したら，きちんと表紙を閉じておかなくてはなりません．これが CLOSE 処理です．

```
CLOSE[[#]ファイル番号 ,
[#]ファイル番号]……
```

INPUT # 文や PRINT # などで入出力の処理をしたあと，その仕事が完了したら必ず CLOSE 文を実行してください．ここで，もしファイル番号のない CLOSE 文を実行すると，オープンしているファイル全てをクローズしてしまうことになります．なお，END 文によっても，自動的に全てのファイルはクローズされます．

5.2 FILES

フロッピィディスクや，バブルカセットなどは，たくさんのデータやプログラムを記録できるので，いったいどんなプログラムやデータが記録されているのかおぼえきれず，また中を見たくても人間の目では見えません．そこで，どんなものが入っているかの目録を，CRT やプリンタに表示させる方法があります．これが FILES というコマンドです．

```
FILES ["デバイス名"][,L]
```

"デバイス名"はファイルディスクリプタのデバイス名で，[,L] を指定すれば，同時にプリンタにも出力されます．

FILES では，ファイルの名前だけではなく，種類，形式，編成，サイズなどを表示してくれます．なおサイズは，そのファイルの占める大きさを表わし，バブルカセットの場合はページ数で，フロッピィディスクの場合は 2 K バイトを 1 とするクラスタ数で示されます．なお，カセットテープの場合は，ファイル名と種類だけが表示され，サイズは出ません．

フロッピィディスクのデバイス 1 に入っているディスケットに，どんな種類のプログラムが入っているかを調べてみましょう．それには FILES "1:" と入れてやればよいのです．もしデバイス 0 に入っているディスケットであれば，単に FILES だけでファイルディスクリプタを省略してもかまいません．

```
FILES"1:"

F3.13-1   O B S 1

151 Clusters Free
```

結果によるとフリーサイズは，151クラスタ，つまりこのディスケットはまだ302Kバイトの余裕があることを示しています．

ここでファイル名はSAVE，OPENで指定したファイルネーム（8字以内）です．種類は0ならBASICのプログラム，1ならデータのファイル，2なら機械語のプログラムです．形式は，AならアスキーコードのファイルBならバイナリコードのファイルであることを示します．ここではバイナリコードです．編成はSならシーケンシャルファイル，Rはランダムファイルです．

そしてサイズは，プログラムやデータの大きさを示しています．ここでは試しに，とても簡単なプログラムを入れてみたため，1クラスタしか使っていません．また，このBASICプログラムが一つしか入っていませんが，もっとたくさん入っていたら，このような表が次々に表示されてくるはずです．

5.3 PRINT

プリンタ

ファイル番号で指定されたファイルに，指定された内容をアウトプットする命令がPRINT #～です．簡単ですからさっそくプログラム例を調べることにしましょう．ただし，アウトプット結果は，この本のサイズ上の問題から，実際のプリントアウトとは，レイアウトが少し違っています．

```
10 OPEN"O",#1,"LPTO:"
20 PRINT #1,"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟ
  ｭｰﾀ ﾌﾟﾘﾝﾄ ｱｳﾄ ﾌﾟﾛｸﾞﾗﾑ"
30 FOR I=32 TO 247
40 PRINT #1,CHR$(I);
50 NEXT I
60 PRINT #1
70 CLOSE #1
```

プログラムのアウトプット結果

```
ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ ﾌﾟﾘﾝﾄ ｱｳﾄ ﾌﾟﾛｸﾞﾗﾑ
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[¥]^_`abcdefghijklmno
pqrstuvwxyz{|}~▁▂▃▄▅▆▇█▏▎▍▌▋▊▉┼┴┬┤├▔─│▕┌┐└┘╭╮╰╯ ｡｢｣､･ｦｧｨｩｪｫｬｭｮｯｰｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿﾀ
ﾁﾂﾃﾄﾅﾆﾇﾈﾉﾊﾋﾌﾍﾎﾏﾐﾑﾒﾓﾔﾕﾖﾗﾘﾙﾚﾛﾜﾝﾞﾟ═╞╪╡◢◣◥◤♠♥♦♣●○╱╲╳円年月日時分秒
```

行番号10で，プリンタ出力をファイル番号＃1に指定しています．そして，行番号20，40のPRINT ＃1で文字列をプリンタで書き出してきます．CHR $(I)は，キャラクタコードIの文字を示すことは，すでにご存知のとおりです．


```
10 OPEN"O",#1,"LPTO:"
20 PRINT #1,"ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ "
30 PRINT #1,"ﾌﾟﾘﾝﾄ ｱｳﾄ ﾌﾟﾛｸﾞﾗﾑ"
40 PRINT #1
50 FOR I=32 TO 247
60 FOR J=1 TO 7
70 IF I=32+28*J THEN PRINT#1
80 NEXT J
90 PRINT #1,CHR$(I);
100 NEXT I
110 PRINT #1
120 CLOSE #1
```

```
ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ
ﾌﾟﾘﾝﾄ ｱｳﾄ ﾌﾟﾛｸﾞﾗﾑ

 !"#$%&'()*+,-./0123456789:;
<=>?@ABCDEFGHIJKLMNOPQRSTUVW
XYZ[¥]^_`abcdefghijklmnopqrs
tuvwxyz{|}~▁▂▃▄▅▆▇█▏▎▍▌▋▊▉┼
┴┬┤├▔─│▕┌┐└┘╭╮╰╯ ｡｢｣､･ｦｧｨｩｪｫ
ｬｭｮｯｰｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿﾀﾁﾂﾃﾄﾅﾆﾇ
ﾈﾉﾊﾋﾌﾍﾎﾏﾐﾑﾒﾓﾔﾕﾖﾗﾘﾙﾚﾛﾜﾝﾞﾟ═╞╪╡
◢◣◥◤♠♥♦♣●○╱╲╳円年月日時分秒
```

```
PRINT #ファイル番号 [, 出力ならび]
```

```
PRINT #ファイル番号, USING
フォーマット文字列; 出力ならび
```

ファイル番号は，前述のOPEN文によってそのファイルを，出力モード "O" または "A" でオープンしているファイルに，データをアウトプットします．前述のプログラム例ではプリンタでした．

ここで，出力ならびというのは，その値がファイルに書き込まれる数値式または文字式のことです．出力ならびが二つ以上あるときは，その間をセミコロンまたはコンマで区切ってください．文字列の例でいうと，セミコロンの場合は続けて，またコンマの場合は9文字置いてプリントアウトされます．

文字データをプリントアウトするときは，その直後に区切り記号として，コンマを出力するようにします．そうでないと，文字データが続いて出力したとき，それらのデータは連続した文字列として出力されるために，後述のINPUT文では一つの文字データとして読み込んでしまいます．なお，"を書き込む場合は，CHR $(34)を使用します．

PRINT #ファイル番号, USING～については，PRINT USING文（4.14）を参照してください．

```
10 CLEAR 400
20 A$="ｼﾞｭｹﾞﾑｼﾞｭｹﾞﾑ"
30 A$=A$+"ｺﾞｺｳﾉｽﾘｷﾚ"
40 A$=A$+"ｶｲｼﾞｬﾘｽｲｷﾞｮﾉｽｲｷﾞｮｳﾏﾂｳﾝﾗｲﾏﾂﾌｳﾗｲﾏﾂ"
50 A$=A$+"ｸｳﾈﾙﾄｺﾛﾆｽﾑﾄｺﾛﾔﾌﾞﾗｺｳｼﾞﾉﾌﾞﾗｺｳｼﾞ"
60 A$=A$+"ﾊﾟｲﾎﾟﾊﾟｲﾎﾟﾊﾟｲﾎﾟﾉｼｭｰﾘﾝｶﾞﾝｼｭｰﾘﾝｶﾞﾝﾉｸﾞｰﾘﾝﾀﾞｲ"
70 A$=A$+"ｸﾞｰﾘﾝﾀﾞｲﾉﾎﾟﾝﾎﾟｺﾋﾟｰﾉﾎﾟﾝﾎﾟｺﾅｰ"
80 A$=A$+"ﾉﾁｮｳｷｭｳﾒｲﾉﾁｮｳｽｹ"
90 OPEN"O",#1,"LPT0:"
100 PRINT#1,LEFT$(A$,50)
110 PRINT#1,MID$(A$,51,50)
120 PRINT#1,MID$(A$,101,50)
130 PRINT#1,MID$(A$,151)
140 CLOSE#1
```

```
ｼﾞｭｹﾞﾑｼﾞｭｹﾞﾑｺﾞｺｳﾉｽﾘｷﾚｶｲｼﾞｬﾘｽｲｷﾞｮﾉｽｲｷﾞｮｳﾏﾂｳﾝﾗｲﾏﾂﾌｳﾗ
ｲﾏﾂｸｳﾈﾙﾄｺﾛﾆｽﾑﾄｺﾛﾔﾌﾞﾗｺｳｼﾞﾉﾌﾞﾗｺｳｼﾞﾊﾟｲﾎﾟﾊﾟｲﾎﾟﾊﾟｲﾎﾟﾉｼｭ
ｰﾘﾝｶﾞﾝｼｭｰﾘﾝｶﾞﾝﾉｸﾞｰﾘﾝﾀﾞｲｸﾞｰﾘﾝﾀﾞｲﾉﾎﾟﾝﾎﾟｺﾋﾟｰﾉﾎﾟﾝﾎﾟｺﾅｰ
ﾉﾁｮｳｷｭｳﾒｲﾉﾁｮｳｽｹ
```

```
10 REM ** ｴﾝ ﾉ ﾒﾝｾｷ ｹｲｻﾝ **
20 INPUT"ﾊﾝｹｲ ｦ ｲﾚﾖ ";R
30 S=R*R*3.14159
40 OPEN"O",#1,"LPT0:"
50 PRINT"ﾊﾝｹｲ";R;"ﾉ ﾒﾝｾｷ ";S
60 PRINT#1,"ﾊﾝｹｲ";R;"ﾉ ﾒﾝｾｷ ";S
70 CLOSE #1
80 PRINT
90 GOTO 20
ﾊﾝｹｲ 50 ﾉ ﾒﾝｾｷ  7853.98
ﾊﾝｹｲ 3 ﾉ ﾒﾝｾｷ  28.2743
ﾊﾝｹｲ 83 ﾉ ﾒﾝｾｷ  21642.4
```

◆ 1日1割の利子

お正月に，電話をかけるため10円借りました．そしてよせばいいのに，おとそ気分で1日1割の利息を付けることにしたのです．もちろん複利です．60日後，ふとそのことを思い出し，計算してみることにしました．

お正月が10円，1日目が11円，2日目が12.1円……．この辺まではわかりますが，これが60回続くとどうなるでしょう．1日経過するごとに，小数点以下が1桁ずつふえていく計算になるはずですから，小数点以下が59桁になるのは間違いありませんね．次に小数点以上は 10×1.1^{60} ですから，電卓を使ってもほぼ3044.……となり，4桁になることははっきりしています．でも正確な値になると，もちろん倍精度変数を使っても追いつきません．

こんなとき便利なのが配列変数です．1桁1変数とし，この変数の値が10以上になったら，1桁上の変数に1をたしてやる考え方です．

また，1.1倍するということは，数値を1桁分ずらしてやり，加算することになります．この処理をする行で，変数CYを使っていますが，このような使い方をキャリーフラグといいます．10を超えれば，A(J)に1を加え，超えなければ0を加えることになるわけです．

結果は，プリンタでも印刷し出してみましょう．このプログラムの詳細は，特に説明の必要はありませんね．

```
10 DEFINT A-Z
20 DIM A(66)
30 A(63)=1:A(62)=0
40 FOR I=1 TO 60
50 FOR J=1 TO 65
60 A(J)=A(J)+A(J+1)+CY
70 IF A(J)>=10 THEN CY=1:
   A(J)=A(J)-10 ELSE CY=0
80 NEXT J
90 NEXT I
100 REM ｹｲｻﾝ ｹｯｶ PRINT OUT
110 OPEN "O",#1,"LPTO:"
120 FOR I=65 TO 1 STEP-1
130 PRINT USING "#";A(I);
140 PRINT#1,USING"#";A(I);
150 IF I=62 THEN PRINT".";:
    PRINT#1,".";:
160 NEXT I
170 PRINT
180 PRINT#1
190 END
```

行番号60……Jが1だけ前のA(J)を加える．もしCYが1ならそれも加える．

例　1210
　＋ 121
　　1331　⇨ 1210×1.1 =1331

行番 70……A(J)が2けたになったらCYを1にする．そしてA(J)から10を引き1けたに直してやる．

```
3044.8163954141809957444929536027877463903841506669808862194760100
```

これをRUNさせると，ちょっと演算に時間がかかります．結果が出るまでしばらく待ってください．

5.4 INPUT#, LINE INPUT#

ファイルからデータを読み込み，変数に代入します．

> **INPUT#**
> **ファイル番号，変数名［，変数名］**

> **LINE INPUT#**
> **ファイル番号，文字変数**

OPEN文で，モードが"I"でオープンしているファイルからデータを読み取ります．変数は，読み取ったデータを格納しておく入れ物です．したがって読み取るデータと変数名の型は，当然一致していなくてはなりません．データの区切りには次の文字が使えます．

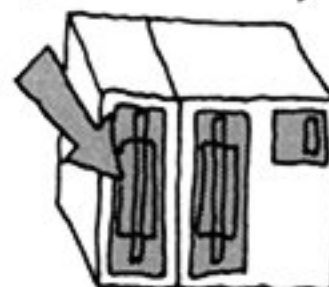
数値の場合：空白，コンマ，コロン，CR，LF
文字列の場合：コンマ，コロン，CR，LF

なお文字列の場合，データの最初が"ならば，次の"までが読み込まれる文字データです．したがって，INPUT#～の場合，"で囲まれた文字列は"を文字として含むことはできません．

LINE INPUT#文の場合は，CRコードまでを文字変数に読み込みます．読み込める文字数は最大255文字です．

では，実際のプログラムを調べてみましょう．まずフロッピィディスクにデータを書き込んでおかなくてはなりません．それにはまず"O"でOPENする必要があります．次に，パーソナルとコンピュータと書き込んでみましょう．

```
0 REM PRINT#
10 CLS
20 OPEN "O",#1,"1:ﾃﾞｰﾀ"
30 PRINT "ﾊﾟｰｿﾅﾙ"
40 PRINT #1,"ﾊﾟｰｿﾅﾙ"
50 PRINT "ｺﾝﾋﾟｭｰﾀ"
60 PRINT #1,"ｺﾝﾋﾟｭｰﾀ"
70 CLOSE #1
80 END
```

これをRUNさせると，ドライブ1の方のフロッピィディスクに，ファイル名データという二つの文字列からなるデータが書き込まれたわけです．あとはNEWにより，このプログラムを忘れさせてもかまいません．もしいつまでもプログラムを記録しておきたかったら，SAVE"1:データ"などにより，SAVEしておいてください．

では，今度はコンピュータの方にこのデータを読み取ることにしましょう．二つの文字列の変数をA＄，B＄とします．

```
10 CLS
20 OPEN "I",#1,"1:ﾃﾞｰﾀ"
30 INPUT#1,A$
40 PRINT A$
50 INPUT#1,B$
60 PRINT B$
70 CLOSE #1
80 END
```

今度は，行番号20でモード″I″とし，OPENしなくてはなりません．

行番号30で，A＄のファイルに記録されている先頭のデータが，行番号50でB＄に次のデータが入力されます．そして同時に，行番号40, 60によりCRT上にもA＄，B＄の内容が表示されて出てくるはずです．

ここでは，ごく簡単な例を述べましたが，もちろんこのようにして読み出したデータを，今まで述べてきたようないろいろな加工処理によって手を加え，違ったものに仕立て上げることも，さらにOPEN″O″，＃2，″LPT0：″によってプリントアウトしてやることも，それはあなたの自由というわけです．

```
ﾊﾟｰｿﾅﾙ
ｺﾝﾋﾟｭｰﾀ
```

では別の例として，数値変数の場合を調べてみましょう．配列変数X(1)～X(5)に1，2，3，4，5の値を2倍した数値を代入し，スウチという名前のファイルに記録します．

```
10 CLS
20 DIM X(5)
30 OPEN "O",#1,"1:ｽｳﾁ"
40 FOR I=1 TO 5
50 Y=I*2:X(I)=Y
60 PRINT#1,X(I)
70 NEXT I
80 CLOSE #1
90 END
```

今度は，このデータをZ(1)～Z(5)に代入して，その数をプリンタで印刷し出し，同時にその数だけ＊印を横に並べてみます．行番号80～100が＊印を横に並べるプログラムです．なお，行番号110は，行を変えるためのプログラムです．

```
10 CLS
20 DIM Z(5)
30 OPEN "I",#1,"1:ｽｳﾁ"
40 OPEN "O",#2,"LPTO:"
50 FOR I=1 TO 5
60 INPUT#1,Z(I)
70 PRINT#2,Z(I),
80 FOR J=1 TO Z(I)
90 PRINT#2,"*";
100 NEXT J
110 PRINT#2
120 PRINT Z(I)
130 NEXT I
140 CLOSE #1,#2
150 END
```

これをRUNさせ，プリンタに打ち出された結果は次のようになります．

```
2             **
4             ****
6             ******
8             ********
10            **********
```

ここで示したファイルの方法は，シーケンシャルファイルといい，データを続けてインプット，またはアウトプットする方法です．シーケンシャルファイルの場合，その一部のみを変更してやったりすることはできません．このような場合には，次の章で述べるランダムファイルに頼らなくてはなりません．

◈ EOF（ファイル番号）

シーケンシャルファイルの場合の，終わりを検出する関数です．End of Fileの略ですね．ファイルが終わりになると，－1（真）となりますが，そうでない場合には0（偽）を与えます．

たとえば，IF文によって，－1か0かを検出し，－1になったら必要場所にジャンプさせるときなど，便利な関数として利用できます．

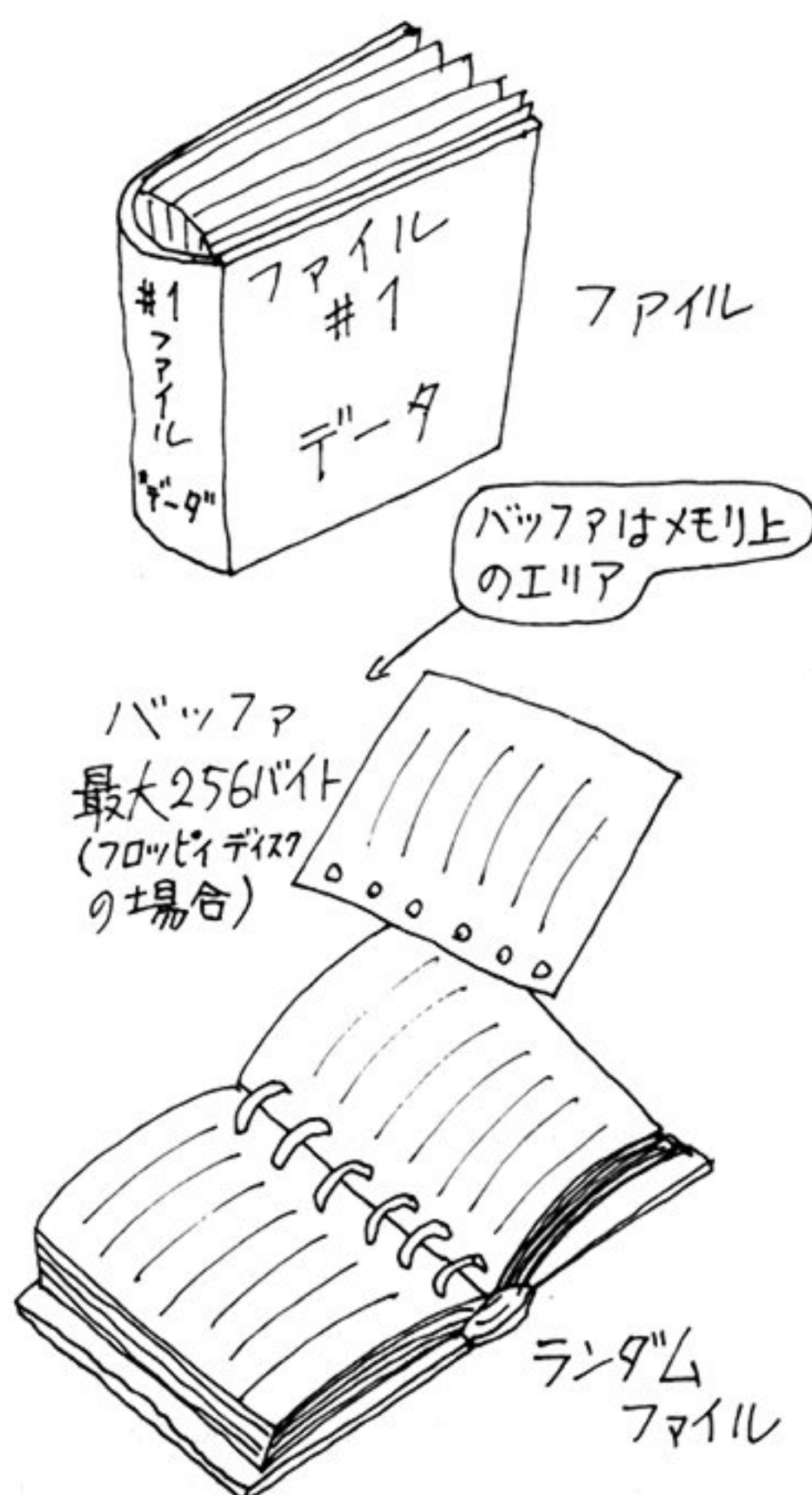
5.5 ランダムファイル

フロッピィディスクやバブルメモリは，連続的な，つまりシーケンシャルな使い方しかできないカセットテープと違い，自分の好きなところにデータを書き込んだり，あるいは読み出したりすることのできるたいへん便利な外部記憶装置です．この便利な道具を自由に使いこなすためには，実はここに述べる程度ではいいつくせない部分が少なくありません．

ここでは，このランダムファイルについて，その基本的な命令の使い方について触れてみたいと思います．ランダムファイルを自由に使いこなしてこそ，初めてコンピュータの真価が発揮できるわけですから，別の機会を得て，その詳細な使い分けについて学ばれることを希望します．

◆ OPEN"R"#ファイル番号，"ファイルディスクリプタ"

5.1で示したように，モードのか所にはRを入れます．ランダムファイルは，どこからでも読み書きができるので，I，Oなどの区別は必要ないのです．シーケンシャルファイルの場合は，データの追加（アペンド，A）しかできません．

シーケンシャルファイルをたとえていうと，昔の書物である巻き物に相当します．そのために，途中に別のデータを書き込んだりするわけにはいかないのです．これに対し，ランダムファイルは，ルーズリーフ式のファイルに当たります．つまり1ページ分の文章を，ファイルの指定した場所に自由に閉じ込むことができるわけです．なおランダムファイルの場合，数値ではなく，文字変数しか扱えないことにご注意ください．また，特にファイル番号0は，後述のDSKI$，DSKO$のときに使用されます．

◆ **FIELD［#］ファイル番号，フィールド幅 AS 文字変数名［，フィールド幅 AS 文字変数名］……**

ランダムファイルのバッファに変数の領域を割り当てる役目をします．ここでバッファというのは，上述のルーズリーフの用紙1枚，つまり1ページ分に当たるわけです．ただし，ルーズリーフ用とは違ってバッファの場合は256バイトと決められています．つまり，256字詰の原稿用紙1枚に当たると考えてもよいのです．

もちろん，256バイト以下で利用してもよいのですが，バッファは256バイトと決められていますから，あとは何も書き込まれないことになります．ちょうど1枚の原稿用紙に何文字書こうと1枚は1枚，と同じことです．

```
FIELD#1, 8 AS A$, 6 AS B$
```

この例では，最初の8文字を文字変数A $用とし，次の6文字を変数B$用として，つまり合計14文字しか使っていないことになり，たいへんぜいたくな使い方です．なお，もし256バイトを超えて合計のフィールド幅を指定すると，Field Over flowのエラー表示が出ます．また，それぞれの変数に対して同じフィールド幅を指定することができる場合には，たとえば，

```
FOR I=0 TO 10
FIELD#1, 8AS A$(I), 6AS B$(I)
NEXT I
```

のように，まとめて，領域を割り当てたいところですが，一つのFIELD文で割り当てる必要がありますので，

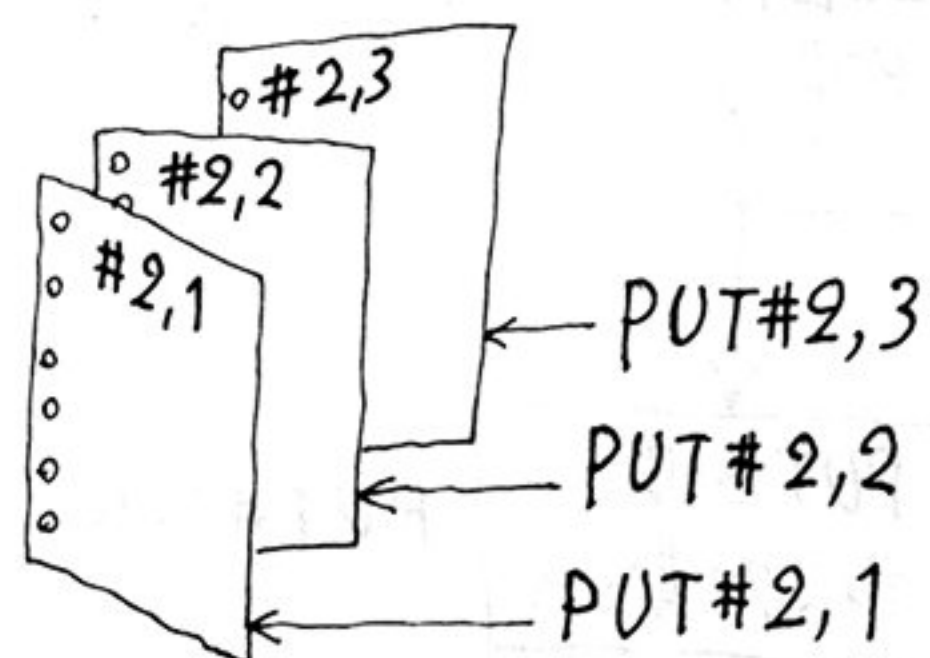
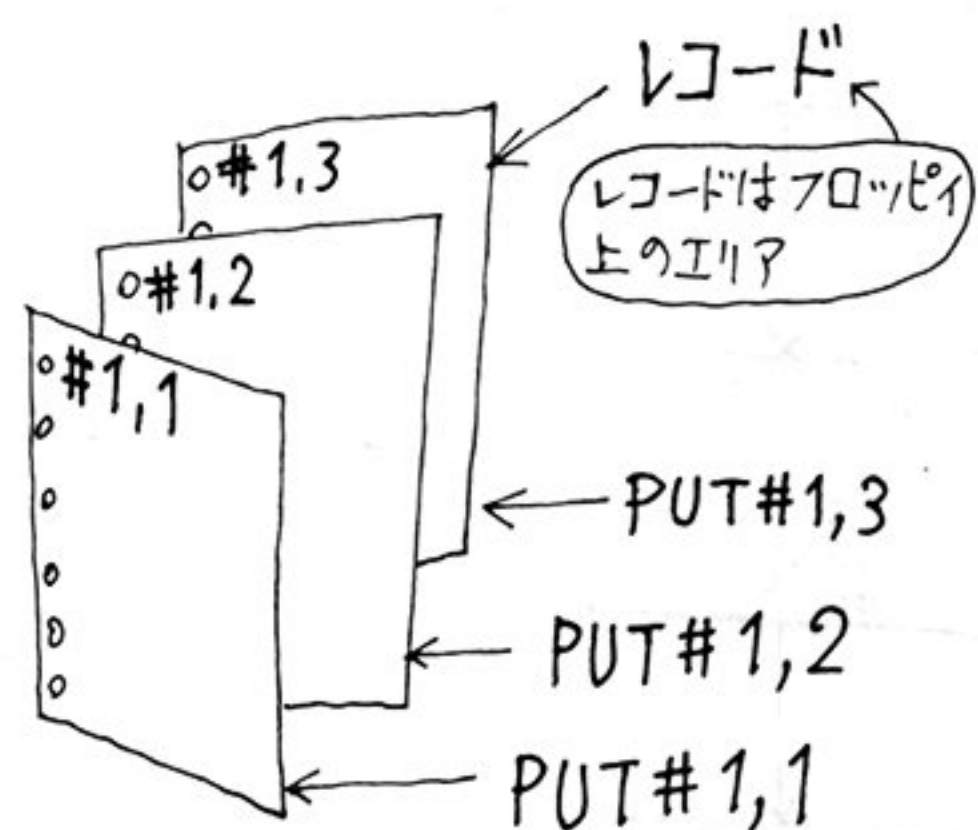
```
FIELD#1 8 AS A$(0),…,8 AS A$(10)
        ,6 AS B$(0),…,6 AS B$(10)
```

のように指定します．

◆ **PUT［#］ファイル番号［，レコード番号］**

ルーズリーフ式のランダムファイルの，何ページ目に閉じ込むかを示すレコード番号を割り当てて，閉じこんでやるための命令です．この命令は，単に閉じ込んでやるだけの機能しかありませんから，原稿用紙に書き込むのに当たる仕事は，次に示すLSET, RSETをあらかじめ実行しておかなくてはなりません．

レコード番号，つまりページ数は何ページ目から作っていってもよいのですが，たとえば200と入れてやると，それだけのレコードが準備されてしまい，それだけで256×200バイト使うことになります．そして，そのためにエラーメッセージDisk Fullが出て，ディスクが一杯になり，データの登録ができないということにもなりかねません．

ランダムファイルを作るときは，1ページ目から順にPUTしていくのがもっとも妥当なやり方でしょう．そして，このように順序正しくPUTするときは，レコード番号を省略することができます．

◆ LSETまたはRSET　文字変数＝式

ランダムファイルバッファ，つまり原稿用紙に書き込みをするのがこの命令です．ここでいう文字変数は，もちろんFIELD文で割り当てられた変数名のことです．そして，これらは文字変数ですから，文字変数に代入される式もまた文字式でなくてはなりません．

式の結果の文字列の長さが，文字変数のフィールドの長さよりも短い場合，

LSET文ではそのフィールドに左詰めで，

RSET文では右詰めで

データを満たしていきます．そして残った部分があれば，その文字の部分には空白が入ります．また式の方が長い場合は，右側から文字が捨てられてしまいます．

また，数値をLSET, RSETで入れてやりたい場合には，このままでは入れることができないので，次のような関数を使って文字形式に変えてやる必要があります．

◆ MKI$（整数表記）
MKS$（単精度表記）
MKD$（倍精度表記）

数値を文字型に変えてやるには，STR$(X)というのがありました．しかしこれを使うと，数値の状態によって長さがいろいろ異なってくることがあります．ランダムファイルの場合，FIELD文でフィールド幅が決められるため，いつも同じ長さを持つ方が便利です．そこで利用されるのが，このような三つの関数です．

MKI $(式)…… 2バイトの文字列

MKS$(式)…… 4バイトの文字列

MKD$(式)…… 8バイトの文字列

◆ CVI, CVS, CVD

前述の文字式から，再び数値に戻すときに使われる関数です．たとえば，MKI$を使って文字に直したら，あとは必ずCVIを使ってもとに戻す必要があります．では，このお互いに逆関数の関係にある二種類の関数を使って，プログラムを作ってみましょう．

```
10 A%=12345
20 B!=123456
30 C#=1234567890
40 A$=MKI$(A%)
50 B$=MKS$(B!)
60 C$=MKD$(C#)
70 PRINT CVI(A$)
80 PRINT CVS(B$)
90 PRINT CVD(C$)
RUN
 12345
 123456
 1234567890
```

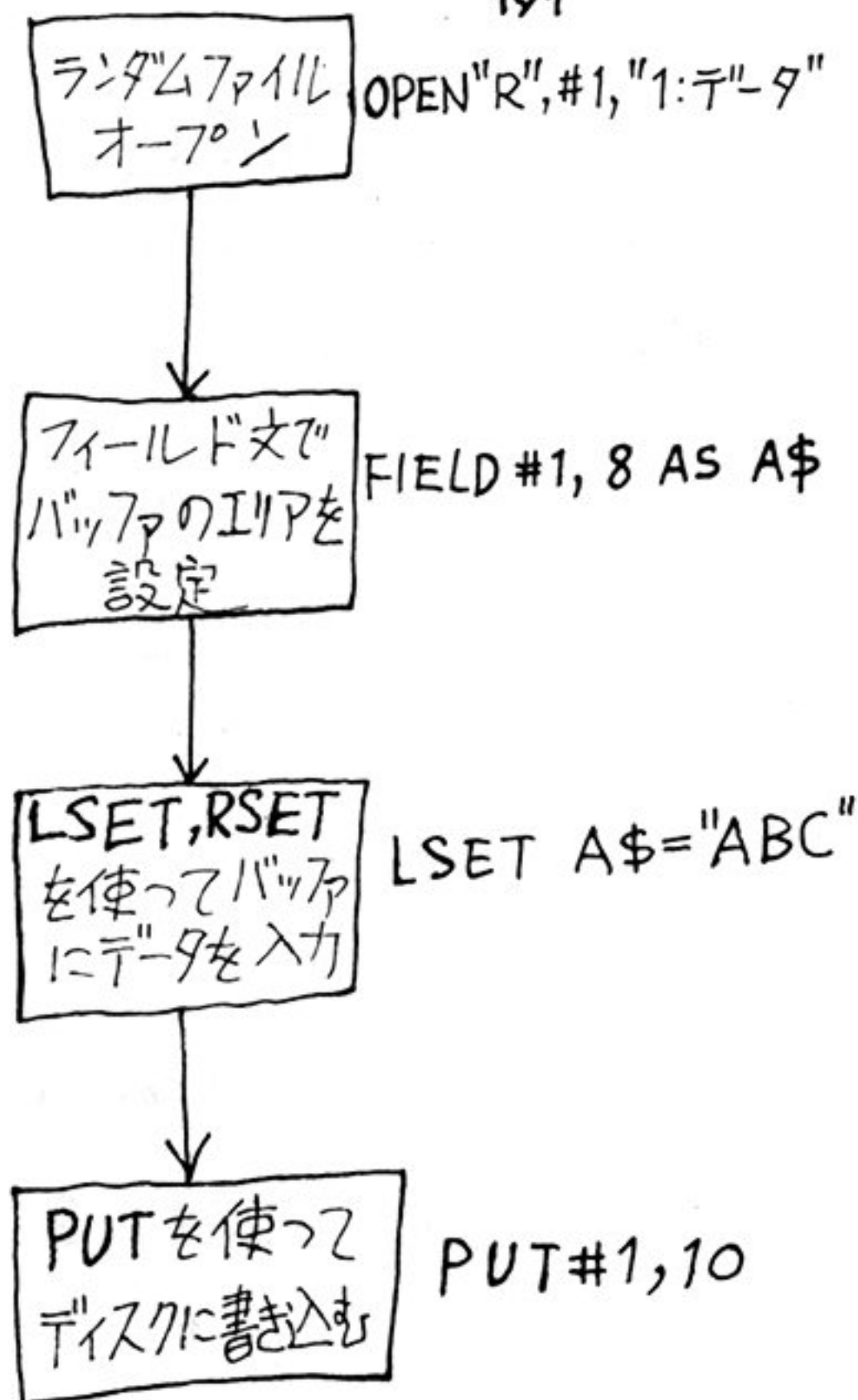
FIELD#1, 5 AS A$, 5 AS B$

LSET A$ = "アイウ"
RSET B$ = "カキク" ｝なら

A$ → アイウ
B$ → 　　カキク

LSET A$ = "アイウエオカキクケコ"
RSET B$ = "サシスセソタチツテト" ｝なら

A$ → アイウエオ

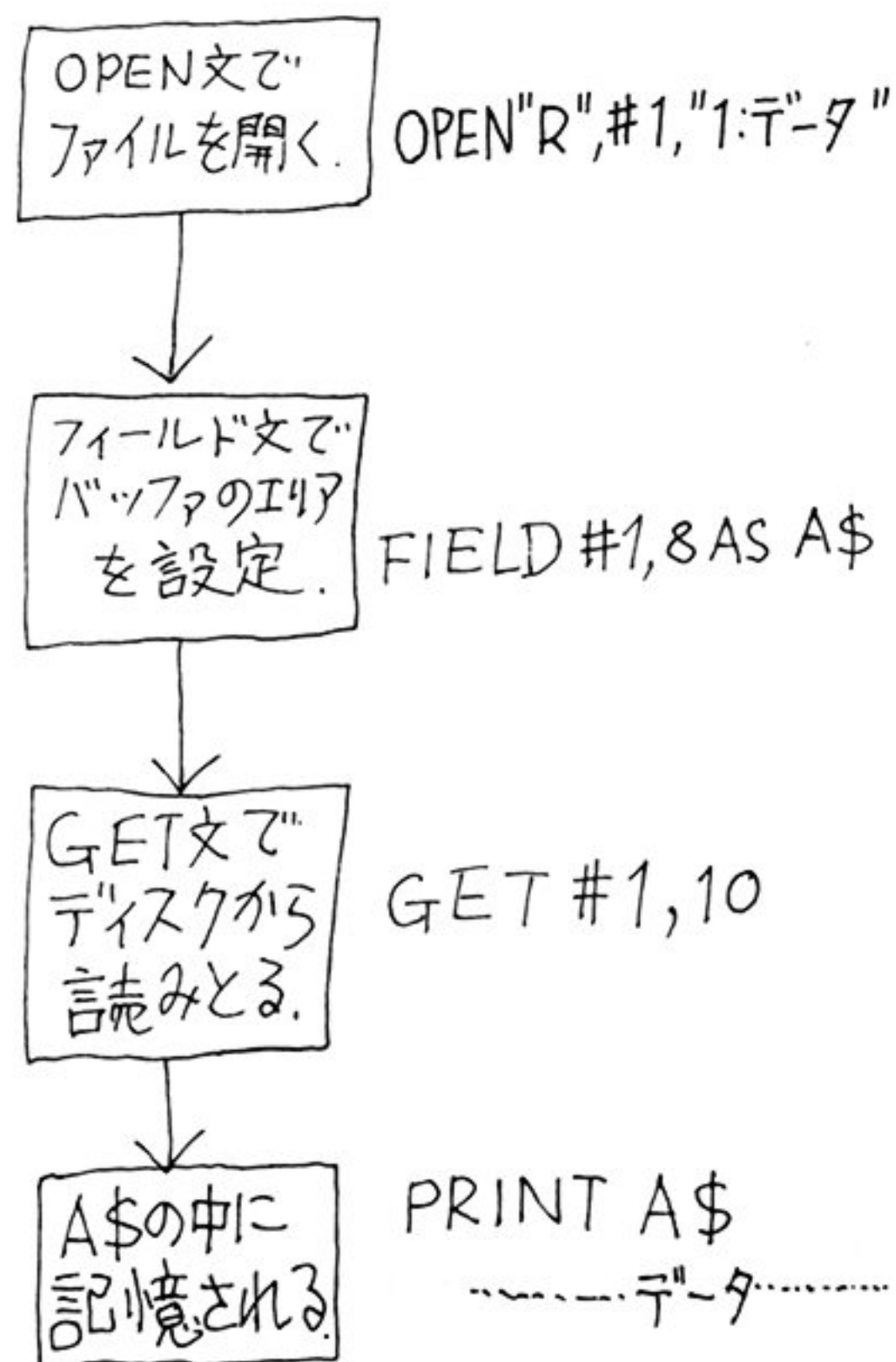
B$ → サシスセソ

◆ GET [#] ファイル番号 [, レコード番号]

ランダムファイルから，コンピュータに読み込むときに使うのが，このGETという命令です．PUTの逆ですね．PUTは1レコード，つまりルーズリーフでいうと，1ページ分を閉じ込んでやる仕事をしましたが，GETは逆に1ページ分を取り出す役割を果たします．たとえば，

GET#1，10

では#1のファイルから，10ページ目を取り出せ，という意味です．PUTの場合も，その中身がどうであるかなどといったことは関与しなかったのと同様，GETではただ1ページ分取り出す仕事しかしません．このような仕事を引き受けるのはFIELD文です．たとえば，

FIELD#1，8 AS A$，6AS B$

とあると，先ほどの10ページ目の先頭の8文字がA$に，そして次の6文字がB$に入ることになるわけです．

◆ LOF (ファイル番号)

ファイル番号で示される入力ファイルの，バッファに入っている文字数を与えます．

◆ LOC (ファイル番号)

ファイル番号で指定されるファイル上で，直前にGETまたはPUTされたレコードの，次のレコード番号を与えます．

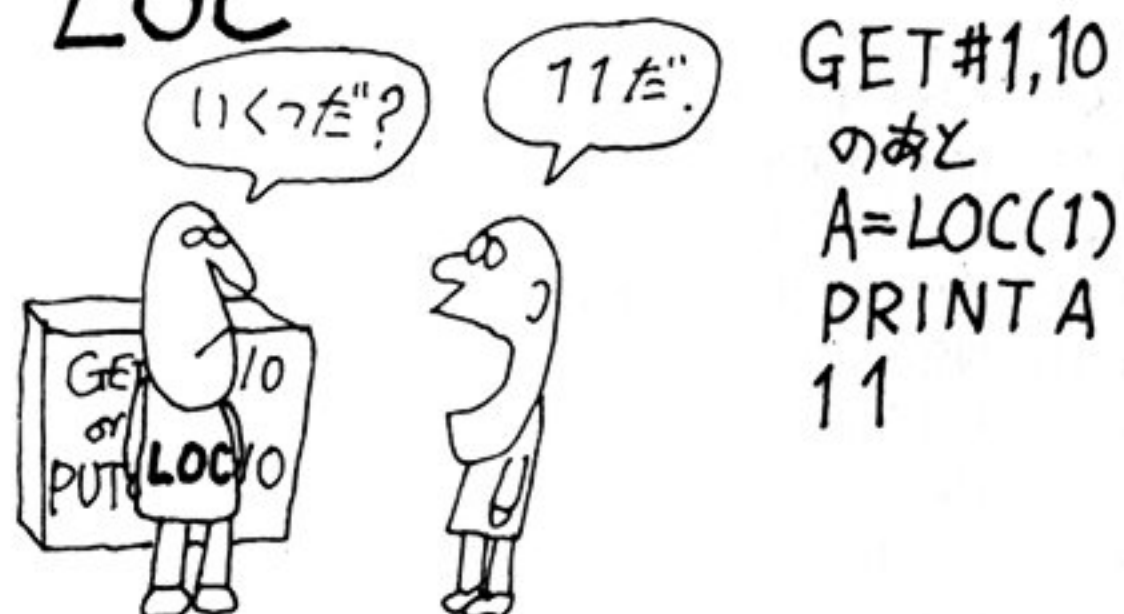
◆ DSKI$またはDSKO$ (ドライブ番号，トラック番号，セクタ番号)

DSKI$は，ディスケット上の指定されたセクタの内容を直接読み取る場合，またDSKO$は，直接書き込みをする場合のための関数です．

DSKI$関数を使用する前には，FIELD文で，0番バッファを使って文字変数を定義しておかなくてはなりません．

FIELD#0，40 AS A$，64 AS B$

なお，ドライブ番号は0～3，トラック番号は0～39，セクタ番号は1～32であり，17～32を指定したときは，ディスケットの裏面のセクタ1～16を示します．

DSKO$は，ファイルの構成には全く無関係に直接書き込むわけで，通常の使い方ではないので，詳細については省略します．

◆ ランダムファイルの簡単なプログラム例を調べてみましょう．

```
10 OPEN "R",#1,"1:ランダ゛ム"
20 FIELD #1,10ASA$,10ASB$
30 LSET A$="12345"
40 RSET B$="67890"
50 PUT#1,1
60 CLOSE #1
```

行番号10でランダムファイルをファイル番号＃1，ファイル名ランダムで定義します．行番号20でA$，B$を書き込むことを指定，行番号30，40でランダムファイルバッファに移します．

A$＝″12345……………″

B$＝″……………67890″

行番号50でレコード1にA$，B$をディスク上に書き込みます．

次は読み出しのプログラムです．

```
10 OPEN "R",#1,"1:ランダ゛ム"
20 FIELD #1,10ASA$,10ASB$
30 GET#1,1
40 PRINT A$
50 PRINT B$
60 CLOSE #1
```

これを実行すると次のようになります．

```
RUN
12345
          67890
```

◆ **シーケンシャルファイル例**

```
0 REM シーケンシャル ファイル カキコミ
10 OPEN "O",#1,"ククノデ゛ータ"
20 FOR I=1 TO 9
30 PRINT  I;"ノ ダ゛ン"
40 PRINT #1,I
50 PRINT #1,"ノ ダ゛ン"
60 PRINT
70 FOR J=1 TO 9
80 PRINT I;"*";J;"=";I*J
90 PRINT#1,I
100 PRINT#1,"*"
110 PRINT#1,J
120 PRINT#1,"="
130 PRINT#1,I*J
140 NEXT J
150 PRINT
160 NEXT
170 CLOSE #1
180 END
```



```
9 ノ ダ゛ン

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81

Ready
```

```
0 REM シーケンシャル ファイル ヨミダシ
10 OPEN "I",#1,"ククノデータ"
20 CLS
30 LOCATE 2,2
40 PRINT "フロッピイ ヨリ データ ヲ ヨミトリマス"
50 LOCATE 4,4
60 PRINT "クク ノ ドノ ダン ヲ ヨミマスカ";
70 INPUT I
80 INPUT #1,A,A$
90 IF I=A THEN 150
100 FOR N=1 TO 9
110 INPUT #1,B,B$,C,C$,D
120 NEXT
130 IF EOF(1) THEN 300
140 GOTO 80
150 PRINT
160 PRINT TAB(6);A;A$
170 FOR N=1 TO 9
180 INPUT #1,B,B$,C,C$,D
190 PRINT TAB(8);B;B$;C;C$;D
200 NEXT
210 CLOSE #1
220 END
300 CLOSE #1
305 PRINT
310 PRINT "          データ ナシ"
320 END
```

行番号100〜120はデータを読みとばすプログラム，もしデータの終りまできたらEOFで行番号300に進み，そうでなければ行番号140でつぎのデータを読む．

行番号70で変数Iに1〜9までの数をインプットさせる．

行番号150〜200がCRT上に表示させるプログラム．

表示するデータは行番号90でI=Aの場合．

ファイル名は書き込んだときの名前と同じでなくてはならない．

```
フロッピイ ヨリ データ ヲ ヨミトリマス

  クク ノ ドノ ダン ヲ ヨミマスカ? 9

     9 ノ ダン
       9 * 1 = 9
       9 * 2 = 18
       9 * 3 = 27
       9 * 4 = 36
       9 * 5 = 45
       9 * 6 = 54
       9 * 7 = 63
       9 * 8 = 72
       9 * 9 = 81
```

Iに9をインプットした場合の例

```
フロッピイ ヨリ データ ヲ ヨミトリマス

  クク ノ ドノ ダン ヲ ヨミマスカ? 10

     データ ナシ
```

Iに1〜9以外をインプットした場合（I=Aが見当らない場合）．

◆ ランダムファイルの例

```
0 REM ﾗﾝﾀﾞﾑ ﾌｧｲﾙ ｶｷｺﾐ
10 WIDTH 40,25
20 N=1
30 OPEN "R",#1,"ｱﾄﾞﾚｽ"
40 REM FIELD ｼﾃｲ
50 FIELD #1,10 AS N1$,10 AS N2$,35 AS AD$,6 AS PN$,15 AS TL$
60 REM DATA STR
70 CLS
80 READ A$:LSET N1$=A$
90 LOCATE 2,2:PRINT A$
100 READ A$:LSET N2$=A$
110 LOCATE 13,2:PRINT A$
120 READ A$:LSET AD$=A$
130 LOCATE 2,4:PRINT A$
140 READ A$:LSET PN$=A$
150 LOCATE 2,6:PRINT A$
160 READ A$:LSET TL$=A$
170 LOCATE 2,8:PRINT A$
180 FOR I=1 TO 10
190 NEXT
200 LOCATE 2,17
210 PRINT "ﾌﾛｯﾋﾟｨ ﾃﾞｨｽｸ ﾍ ｶｷｺﾐ ﾏｽ "
220 LOCATE 2,19
230 PRINT "ﾄﾞﾗｲﾌﾞ 0 ﾚｺｰﾄﾞ No";N
240 PUT #1,N
250 FOR I=1 TO 1000
260 NEXT
270 N=N+1
280 IF N>2 THEN LOCATE 2,21:PRINT "ﾃﾞｰﾀ ｶｷｺﾐ ｵﾜﾘ":
      CLOSE #1:END
290 GOTO 60
300 DATA ｽｽﾞｷ
310 DATA ｼｮｳｲﾁ
320 DATA ﾄｳｷｮｳﾄ ｾﾀｶﾞﾔｸ ｻﾝｹﾞﾝｼﾞｬﾔ 1234
330 DATA 154
340 DATA 03-(333)-3333
350 DATA ｻﾄｳ
360 DATA ｴﾂｺ
370 DATA ｵｵｻｶﾌ ﾌｼﾞｲﾃﾞﾗｼ ｺﾔﾏ 0-1-7
380 DATA 583
390 DATA 06-(666)-6666
```

```
ｽｽﾞｷ         ｼｮｳｲﾁ

ﾄｳｷｮｳﾄ ｾﾀｶﾞﾔｸ ｻﾝｹﾞﾝｼﾞｬﾔ 1234

154

03-(333)-3333
```

ﾌﾛｯﾋﾟｨ ﾃﾞｨｽｸ ﾍ ｶｷｺﾐ ﾏｽ

ﾄﾞﾗｲﾌﾞ 0 ﾚｺｰﾄﾞ No 1

ｻﾄｳ　　　　　ｴﾂｺ

ｵｵｻｶﾌ ﾌｼﾞｲﾃﾞﾗｼ ｺﾔﾏ 0-1-7

583

06-(666)-6666

ﾌﾛｯﾋﾟｨ ﾃﾞｨｽｸ ﾍ ｶｷｺﾐ ﾏｽ

ﾄﾞﾗｲﾌﾞ 0 ﾚｺｰﾄﾞ No 2

ﾃﾞｰﾀ ｶｷｺﾐ ｵﾜﾘ

```
0 REM ﾗﾝﾀﾞﾑ ﾌｧｲﾙ ﾖﾐﾀﾞｼ
10 WIDTH40,25
20 N=1
30 OPEN "R",#1,"ｱﾄﾞﾚｽ"
40 REM FIELD ｼﾃｲ
50 FIELD #1,10AS N1$,10AS N2$,35AS AD$,6AS PN$,15 AS TL$
60 REM DATA STR
70 CLS
80 LOCATE 2,17
90 PRINT "ﾖﾐﾀﾞｼﾀｲ ﾚｺｰﾄﾞ No ﾊ";
100 INPUT N
110 GET #1,N
120 LOCATE 2,2:PRINT N1$
130 LOCATE 13,2:PRINT N2$
140 LOCATE 2,4:PRINT AD$
150 LOCATE 2,6:PRINT CHR$(&HF8);" ";PN$
160 LOCATE 2,8:PRINT "TEL ";TL$
170 CLOSE #1
```

ﾖﾐﾀﾞｼﾀｲ ﾚｺｰﾄﾞ No ﾊ? 2

ｻﾄｳ　　　　　ｴﾂｺ

ｵｵｻｶﾌ ﾌｼﾞｲﾃﾞﾗｼ ｺﾔﾏ 0-1-7

〒583

TEL 06-(666)-6666

5.6 その他の操作

ここでは，入出力装置を操作する上で，まだお伝えしていないいくつかの操作命令についてまとめて示します．

◈ DSKINI

システムディスク作成後，フロッピィディスクの初期化を実行します．

DSKINI　ドライブ番号

フロッピィディスクを初期化するときに使用します．

ドライブ番号は，初期化するディスケットをセットしたドライブの番号です．このコマンドを入力したとき，Are You sure（Y or N）？　の問い合わせのメッセージが出ます．そこで，初期化を実行するならY，初期化を中止するならNをキーインしてやれば完了します．

このコマンドを実行すると，今まで書き込まれていたファイルが全て消えるので，注意してください．

◈ BUBINI（FM－8の場合）

バブルカセットの初期化を行ないます．

BUBINI　ユニット番号

バブルカセットの初期化を実行すると，今まで書き込まれている内容は消されます．

ユニット番号は0，または1です．初期化を実行するときは，バブルカセットの裏側のライトプロテクトをRECORD側にしてください．この状態でバブルカセットをセットし，BUBINIを実行し

ます．

このコマンドを入力すると，Are you sure（Y or N）？のメッセージを表示します．そこで，初期化するならYを，中止するならNを入れてやればよいのです．このとき，もしライトプロテクトがRECORD側になっていなければ，"Device I／O Error" と表示して，BASICコマンド待ちの状態に戻ります．なお，BUBINIを実行すると，今まで書き込まれた内容が消えますので注意してください．

◈ MERGE

メモリにあるプログラムと，指定されたファイルのプログラムをまぜ合わせます．

MERGE　"ファイルディスクリプタ"

プログラムとプログラムをつなぐ場合に使います．

ファイルの中のプログラムとメモリ中のプログラムをまぜ合わせます．その場合，同じ行番号のものがあればファイル中の行番号のプログラムが優先し，メモリ上に置きかわります．なお，MERGEを使用できるファイルは，アスキー形式でセーブされたファイルだけです．

サブルーチンのプログラムをファイルにたくさん書き込んでおき，プログラムを作る場合にMERGEを使って，サブプログラムを読み込んで作れば，プログラムをキーボードからインプットするよりらくに作れるでしょうし，また大きなプログラムの場合，何人かで手分けして作ることも可能になります．

◆ KILL

フロッピィディスクや，バブルカセット上のファイルを削除します．

KILL ″ファイルディスクリプタ″

フロッピィディスクやバブルカセットに書き込まれているファイルを削除します．

直接モードで，このコマンドを入力したときは，Are you sure (Y or N)？の問い合わせメッセージが表示されるので，削除するならYをインプット，削除しないならNをインプットします．

```
FILES "1:

FUJITSU    0 B S 1
FUJIYAMA   0 B S 1

150 Clusters Free

Ready
KILL "1:FUJIYAMA"
Are you sure(Y or N)? Y

Ready
FILES "1:

FUJITSU    0 B S 1

151 Clusters Free

Ready
```

これは，フロッピィディスク上のファイルFUJIYAMAを削除した例です．

◆ NAME

フロッピィディスクのファイル名を変更します．

NAME ″旧ファイルディスクリプタ″ AS ″新ファイルディスクリプタ″

フロッピィディスクに書き込まれているファイルの名前を変更したい場合に使用します．

二つのファイルディスクリプタのデバイス名は，同じでなければなりません．

```
FILES "1:

FUJIYAMA   0 B S 1

151 Clusters Free

Ready
NAME "1:FUJIYAMA"
                AS"1:FUJITSU"
Ready
FILES "1:

FUJITSU    0 B S 1

151 Clusters Free

Ready
```

フロッピィ上のファイル FUJIYAMAをNAME文を使ってFUJITSUに変更しました．

◆ LOAD？

カセットテープにプログラムをSAVEする場合，プログラムと共にチェックサムデータが録音されています．そこで，プログラムをLOADするときにプログラムをカウントし，その数とテープ上に録音されているチェックサムデータとを照合する命令です．

LOAD？ [″[CAS 0 :]ファイル名″]

プログラムが正しく読み込まれたかどうかチェックして読み込みます．もし，チェックサムが一致しなければ ″Device I/O Error″ が表示されます．つまり，正確に録音されているもの以外はLOADできませんし，テープにキズなどがあってもやはりLOADできません．なお，″CAS 0：ファイル名″ を省略した場合は，テープに録音されている一番最初のプログラムをLOAD？します．

◆ SKIPF

指定したファイルを読み飛ばします．

SKIPF [″CAS 0 ：ファイル名″]

″CAS 0 ：ファイル名″ を省略した場合は，一番最初に読み込まれたファイルを読み飛ばします．

6 より高度な使い方

この章では，パソコンに通信回線をつないで，遠く離れたところにある別の機械と情報をやり取りする方法や，コンピュータが直接理解できる機械語を利用するときに必要なBASICの命令について，その概要にふれてみたいと思います．なお，このようなより高度な使い方については，ここではとうてい説明しつくすことはできません．詳細については，別の参考資料を参照してください．

コンピュータと電話回線とを結ぶ音響カプラー

6.1 通信回線制御機能

FMシリーズ通信回線を利用して，データやプログラムを，遠く離れたところにある別のパーソナルコンピュータや大型のコンピュータとやり取りできる機能を持っています．このための，インタフェースは5個あり，RS－232C規格となっています．

この機能を利用すると，北海道と九州でパソコンゲームをしたり，全国各地にある支店から，コンピュータのデータを読み取ったりすることもできますし，RS－232C規格によるインタフェースを持ったプリンタや，X－Yプロッタ計測器などをコントロールすることも可能です．

なお，RS－232Cとは，データを直列に出し入れするときのルールを定めた規格です．FMシリーズは，このインタフェースをCOM0：～COM4：まで5回路を持っています．

◆ OPEN

通信回線を用いて外部とデータのやり取りをするには，あらかじめ次の命令文を実行して，形式を決めてやらなくてはなりません．

OPEN ″モード″，［#］ファイル番号，″COMn：(オプション)″

モードは，入力のときは ″I″，出力のときは ″O″ です．

ファイル番号は1～16です．他のOPEN文と同じ番号にならないようにします．

COMnはデバイス名です．COM0：～COM4：で，入出力をするポートを指定します．

オプションはボーレート，ビット長，パリティ，ストップビット数を，CBPSの順に指定します．C，B，P，Sのそれぞれの指定の方法は次のとおりです．

C－ボーレート（データを送り出す速さ）

　F：fastクロック（1／16）

　S：slowクロック（1／64）

　のどちらかを指定します．

B－ビット長（一つのデータの長さ）

　7：7ビット／文字

　8：8ビット／文字

P－パリティ（エラーチェックの指定）

　E：偶数パリティ

　O：奇数パリティ

　N：ノンパリティ

S－ストップビット数（一つのデータの終わりを示すビット数）

　1：1ストップビット

　2：2ストップビット

◆ CLOSE

OPEN文で通信回線に割り当てられたファイルを閉じる命令文です.

CLOSE [[#] ファイル番号 [, [#] ファイル番号] ……]

OPEN文とCLOSE文は, 必ずペアーとなってプログラム中で使用します.

OPEN文で指定したファイルは, 用ずみになったら必ずCLOSE文でファイルを閉じてください.

◆ INPUT

OPEN文で指定された通信回線からデータを入力する命令文です.

INPUT #ファイル番号, 変数名 [, 変数名] ……

同じファイル番号のOPEN文で指定されたRS－232C通信回線用ポートからデータを読み取り, 指定された変数に代入します.

データの読み取りはCR, コンマ(,)またはコロン(:)で区切ってありますが, これらの文字コードは変数に代入されません.

◆ LINE INPUT

INPUT #文と同じように, 通信回線のデータを読み取りますが, LINE INPUT #は1行の文字変数として入力する命令です.

LINE INPUT # ファイル番号, 文字変数名

同じファイル番号のOPEN文で指定されたRS－232C通信回線用ポートから, データを255文字以内の文字を, CRコードが現われるまでを1行として読み取り, 一つの文字変数に代入します. 256文字以上の読み取りはできず, 255文字までを変数に代入して実行を終わります. また, CRコードは変数に代入されません.

◆ PRINT

INPUT #の反対で, OPEN文で指定された通信回線にデータを出力する命令で, 次の2形式があります.

PRINT #ファイル番号 [, 式 {, ;} 式] …]

PRINT #ファイル番号, USING フォーマット文字列 ;式[{, ;}式]…

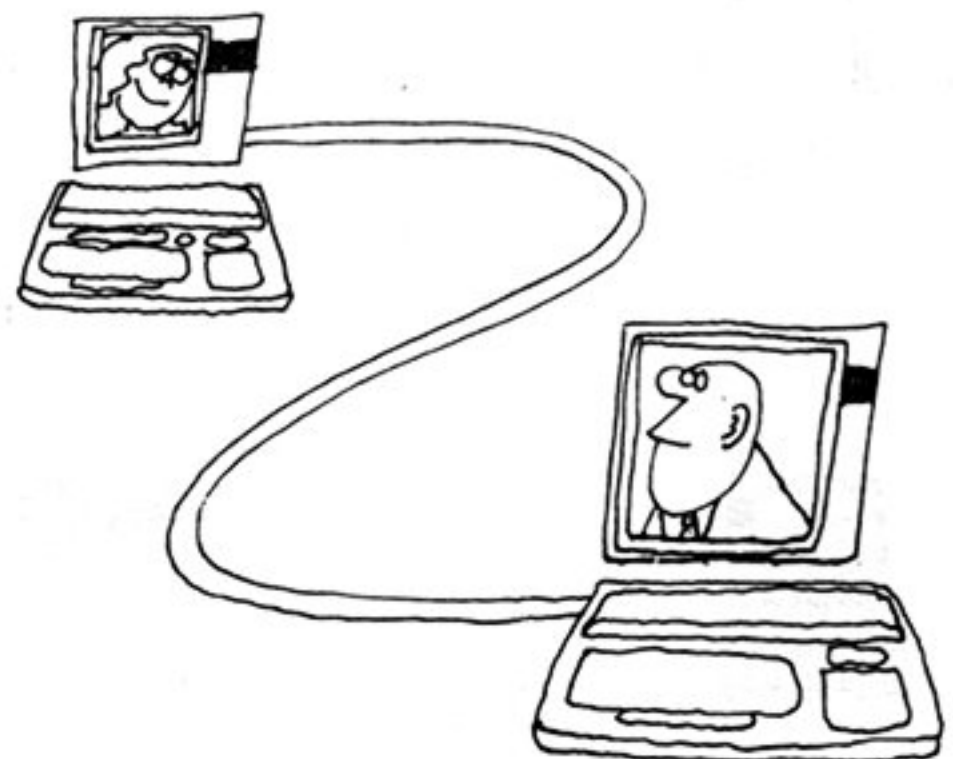
同じファイル番号で, OPEN文で指定されたRS－232C通信回線用ポートからデータを出力します.

USINGフォーマットは, PRINT USING文を参照してください.

INPUT #, LINE INPUTはINPUT, LINE INPUTとほぼ同じで, INPUT文はKEYボードよりデータを読み取り, INPUT #はOPEN文で指定されたRS－232Cポートよりデータを読み取ります.

PRINT #, PRINT # USINGはPRINT, PRINT USINGとほぼ同じで, PRINT文はCRTにデータを表示します. また, RRINT #はOPEN文で指定されたRS－232Cポートよりデータを出力します.

```
OPEN "I", #n, "COMn :(CBPS)"
INPUT #n or LINE INPUT #n
   ⋮
CLOSE #n
OPEN "O", #n, "COMn :(CBPS)"
PRINT #n or PRINT #n USING
   ⋮
CLOSE #n
```

以上のように使用します(nは1から16で指定です).

◆ LIST

通信回線を用いて外部にプログラムリストを出力する命令文です.

```
LIST "COMn：[(オプション)]"
[,[行番号] [{-,}[行番号]]]
```

COMn：で指定されたRS－232C通信ポートへプログラムリストを出力します.

COMnはデバイス名で，COM0：～COM3：のいずれかを指定します.

オプションについては，OPENの項を参照してください．また，行番号の指定はLIST文の説明を参照してください.

この命令文に限り，OPEN文でのファイル指定は必要なく，RS－232Cインタフェイスを持ったプリンタなどに，LISTのプリントアウトする場合に使用します.

◆ ON COM(n) GOSUB

今までの通信回線の入力命令では，データが外部より送られてくるまで，プログラムは停止しています．これではたいへん時間のむだとなり，別の仕事をさせたいときに役に立ちません．そこでデータが送られてきたときだけデータを読み取るように，割り込みをかけるのがON COM(n) GOSUB文です.

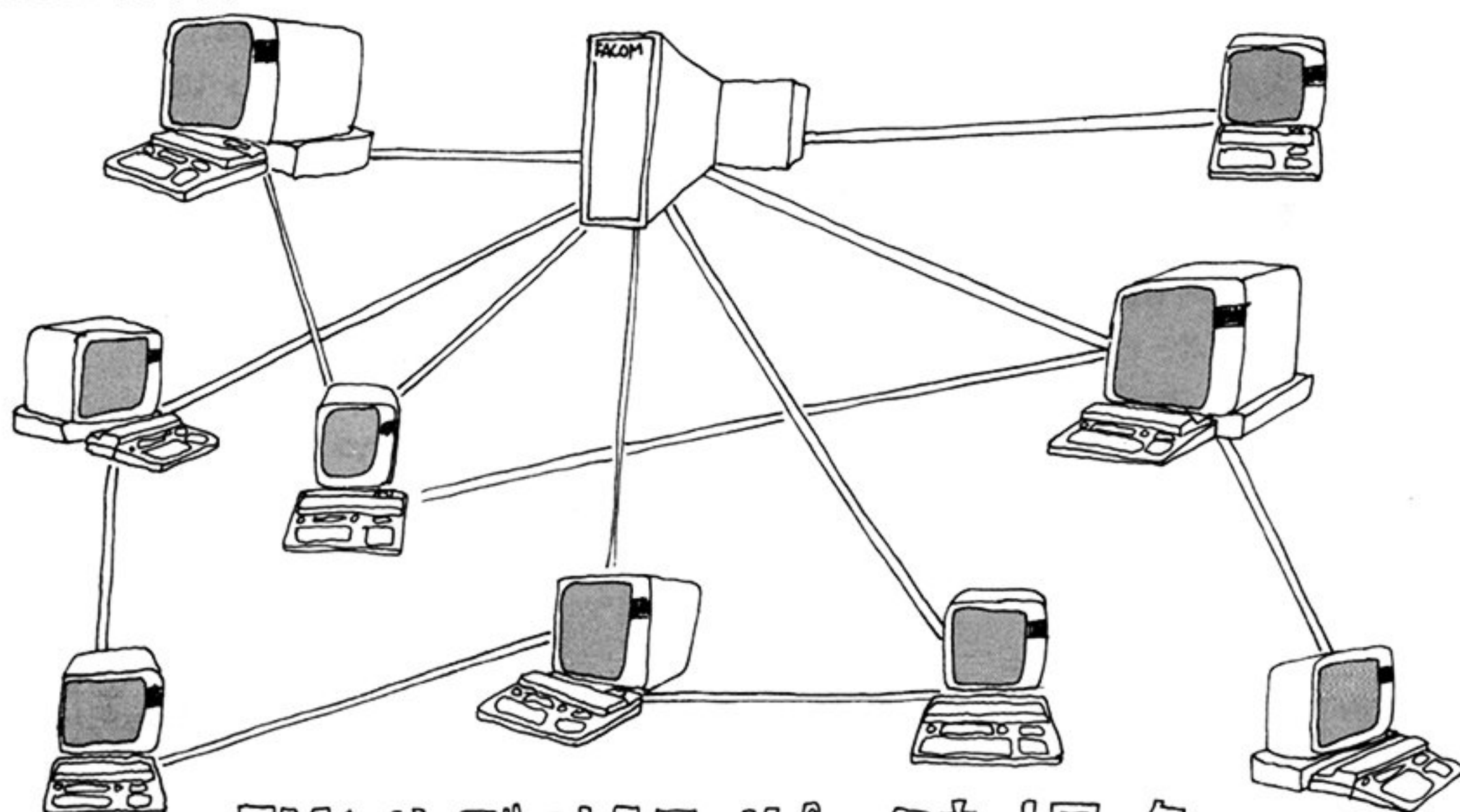
ON COM(n) GOSUB文で入力割り込みをするとき，割り込み実行時のサブルーチンを指定します.

```
          (0)
ON COM {   ∫   } GOSUB 行番号
          (4)
```

行番号は，COM(n)で指定されたポートより，入力があったときに呼び出される割り込み処理ルーチン開始行番号です．また，割り込みルーチンからの復帰は，RETURN文により実行されます.

◆ COM(n) ON／OFF／STOP

ON COM(n) GOSUBで指定した割り込み処理のプログラムを，入力割り込みが発生したときに，割り込みを許可するか，禁止か停止かを指定する命令です.

COM(n)　nはポート番号で0～4の値を取ります.

COM (n)　ONで指定した通信ポートに入力がある場合，ON　COM (n) GOSUBで指定された処理ルーチンを呼び出します．

COM (n)　OFFで指定した通信ポートからの入力割り込みを禁止します．また，COM (n) STOPで指定した通信ポートからの入力割り込みを一時停止し，COM (n) ONで再開します．

ON　COM (n)　GOSUBは，ON　KEY (n)　GOSUBとほぼ同じで，ファンクションKEYよりの割り込みか，通信回線RS－232Cポートよりの割り込みかの違いだけです．

◈ TERM

F－BASICよりターミナルモードに切りかえをする命令です．この命令によりパソコンを端末機として使用できるようになります．

```
TERM [ "cbpsma" ]
```

c…ボーレート

　S：slow　　F：Fast

b…ビット長

　8：8ビット長　　7：7ビット長

p…パリティ

　N：パリティなし　　O：奇数パリティ

　E：偶数パリティ

s…ストップビット

　1：1ストップビット

　2：2ストップビット

m…モード（通信の方法）

　F：全二重通信　　H：半二重通信

a…オートLFを行なうか否かを示す．

　A：オートLFを行なう

　N：オートLFをしない

その他ターミナルモードについては，いろいろな機能があり，注意を要します．F－BASIC文法書，ならびに端末機として使用する場合は，ホスト側のコンピュータの仕様書をよく検討してください．

6.2　機械語

BASICはたいへん便利なプログラム用言語ですが，処理の内容によってはスピードが遅くて困る，という場合もあります．このような場合，速い処理を必要とする部分を機械語で作れば，このような欠点を補うことができます．

◈ POKE

メモリの指定番地にデータを書き込みます．

```
POKE 書き込み番地，データ
```

書き込み番地は，0～65535の範囲であり，データは0～255の範囲です．

◈ PEEK

メモリの中の指定番地のデータを読み出します．つまり，POKEの逆の動きをする命令です．

```
PEEK（式）
```

式は0～65535までの範囲の数であり，アドレスを示します．FMシリーズでは，V－RAM（画面表示用のメモリ）を別のCPUで制御しているため，PEEK，POKEで直接V－RAMの内容を読み出すことはできません．

◈ VARPTR

BASICプログラム中の変数名で指定されるデータが格納されている先頭番地を与えます．

```
VARPTR（変数名）
```

変数名は，任意の型（数値，文字，配列）を使うことができますが，VARPTRを使う前に，その変数には値が代入されていなくてはなりません．この関数によって与えられる番地は，－32768から32767の値を取ります．もし，負の値になった

ときは，実際の値を得るために65536を加えます．

配列に対してVARPTRを使用するときは，全ての単純変数に値が代入されていなくてはなりません．これは新しい変数に値が代入されると，配列のアドレスが変わるためです．

文字変数に対しては，データの格納されているアドレスのポイントを示します．

◈ VARPTRの数値変数を指定した場合

パソコンのMONITORを利用して，メモリをダンプしてみましょう．

```
10 A%=10
20 PRINT VARPTR(A%)
30 PRINT HEX$(VARPTR(A%))
RUN
 2852
B24
```

VARPTRでA%の格納アドレスが&HB24とわかりました．そこでMONを入れて，＊DB24でダンプします．

```
B24  00 0A
     A%の内容
```

Aは整数型変数で2バイトのエリアがあり，このプログラムの場合$(000A)_{16}=(10)_{10}$となります．

```
MON

*DB24

0B24 00 0A 20 30 39 00 00 00
0B2C 42 00 49 52 00 00 00 00
0B34 46 43 41 54 41 4C 4F 47
0B3C 00 00 00 00 30 00 00 00
0B44 00 00 00 00 00 00 00 00
0B4C 00 00 00 00 00 00 00 00
0B54 00 00 00 00 00 00 00 00
0B5C 00 00 00 00 00 00 00 00
*
```

◈ VARPTRの文字変数を指定した場合

前述の場合と同様にして，MONITORでメモリをダンプしてみましょう．

```
10 A$="12345"
20 PRINT VARPTR(A$)
30 PRINT HEX$(VARPTR(A$))

Ready
RUN
 2857
B29
```

VARPTR（文字変数）で，A＄の格納アドレス&HB29とわかりましたので，MON機能で，

```
B29  05        0B 00
     A$の文字数  格納アドレス
```

格納アドレスがB00とわかりましたので，

```
B00  31    32    33
     Aのアスキーコード
```

これを利用して，PEEKで読み取り，POKEで書き込むと，変数の内容を自由に変更することができます．

```
MON

*DB29

0B29 05 0B 00 30 00 00 00 00
0B31 00 00 00 00 00 00 00 00
0B39 00 00 00 00 00 00 00 00
0B41 00 00 00 00 00 00 00 00
0B49 00 00 00 00 00 00 00 00
0B51 00 00 00 00 00 00 00 00
0B59 00 00 00 00 00 00 00 00
0B61 00 00 00 00 00 00 00 00
*

*D0AE0

0AE0 00 00 00 00 00 00 00 00
0AE8 00 00 00 00 00 00 00 00
0AF0 00 00 00 00 00 00 00 00
0AF8 0B 07 00 0A 41 24 E5 22
0B00 31 32 33 34 35 22 00 0B
0B08 14 00 14 B9 20 FF A1 28
0B10 41 24 29 00 0B 25 00 1E
0B18 B9 20 FF 97 28 FF A1 28
*
```

注） これらの値は，BASICの使用状態によりいろいろと異なってくる場合があります．

◆ EXEC

機械語のプログラムを実行する命令です.

EXEC

機械語のサブルーチンを実行する命令ですが, BASICのプログラムと違い, 間違いがあってもエラーの表示をせず, 場合によっては暴走することもあるため, 注意をしなくてはなりません. もしプログラムの間違いにより, BASICに復帰しない場合は, リセットスイッチを押してください.

また, サブルーチンのRETURN命令に相当するのがRTS命令です. なお, レジスタはBASIC側でPUSH, PULLを実行します.

◆ DEF USR

機械語のプログラムの開始番地を指定します.

DEF USR [数字] = [機械語プログラム開始番地]

機械語のサブルーチンを一つの関数として扱えるように, この命令文で定義し, USR+数字が関数名として呼び出すことができます. 数字は0～9までの任意の数です.

機械語プログラム開始番地は, USR [数字] が指定されたときに実行する機械語プログラムの開始番地です.

◆ USR

引数を持ってユーザの機械語のサブルーチンを呼び出します.

USR [数字] (引数)

DEF USRで定義して指定した, 機械語のサブルーチンを実行します.

引数は機械語のサブルーチンにデータを渡すときに必要で, 変数または定数を一つだけ指定できます.

引数情報は, AレジスタとXレジスタに格納し, 機械語サブルーチンに渡されます.

Aレジスタ=2 引数が 整数型
3 〃 文字型
4 〃 単精度実数型
8 〃 倍精度 〃

Xレジスタ=引数が格納されている番地.

①整数型

X→	0	
	1	
	2	上位8ビット
	3	下位8ビット

②文字型

X→	0	文 字 数
	1	文字列が格納さ
	2	れているアドレス

③単精度実数型

X→	0	指 数 部
	1	仮 数 部
	2	
	3	

④倍精度実数型

X→	0	指 数 部
	1	仮 数 部
	2	
	3	
	4	
	5	
	6	
	7	

DEF USR例

```
10    CLEAR , &H5000
20    DEF   USR 1 =&H5000
:                 :
100   A%=USR 1 (A%)
```

DEF USR 1
USR 1 は同番号で指定します.

このUSR関数をうまく使うと, BASICプログラムのデータを機械語サブルーチンへインプット

したり，アウトプットしたりすることができます．

測定器などをパソコンに接続して，データの読み取りは機械語で，測定結果の計算，グラフ表示などはBASICでやれば，簡単にプログラムが作れるでしょう．

◈ LOADM

機械語プログラムをメモリにロードする命令です．

> **LOADM "ファイルディスクリプタ"**
> **[, [オフセット値], [R]]**

機械語で作られたプログラムを，ファイルディスクリプタで指定したファイルよりロードする命令です．オフセット値は機械語のプログラムにプログラムのロードすべき番地が記録されていますが，その番地にオフセット値を加算してロードしたい場合に指定します．Rの指定をすると，プログラムロード後，そのロードしたプログラムを実行します．指定しなければ，プログラムロード後BASICに戻ります．

◈ SAVEM

メモリの内容をファイルにセーブする命令です．

> **SAVEM "ファイルディスクリプタ"**
> **,開始番地，終了番地，入口番地**

ファイルディスクリプタで指定したファイルにメモリの内容をセーブします．

開始番地とは，ファイルすべき内容の先頭番地です．終了番地とは，ファイルすべき内容の終了番地です．

入口番地とは，ファイルした内容のプログラムを実行する場合のスタート番地です．LOADM，EXECコマンドでは，この入り口番地よりプログラムの実行を開始します．

6.3 MONITOR機能

FM−Xは，BASICのプログラム以外に，簡単な機械語がプログラムできるように，モニタ機能が内蔵されています．BASICのコマンドモードより，モニタコマンドを読み出してみましょう．

◈ MON

MONのコマンドによって，BASICのコマンドモードよりモニタコマンドモードに移ります．

> **MON**

BASICのコマンドモードより，モニタコマンドモードに移ります．モニタモードになると，プロンプト"*"を表示し，モニタコマンド待ちになります．

モニタコマンド表

コマンド名	機　　能
M	メモリの内容を変更する
G	指定アドレスに分岐する
R	レジスタの内容を表示．変更もできる
D	指定アドレスより64バイトの内容を表示

モニタモードよりBASICモードに帰るには，STOP，CTRL−C，CTRL−Xキーを入力することにより，BASICモードに戻ります．

◈ M

メモリの内容を表示変更します．

> **M [アドレス]**

アドレスは，1～4桁の16進数でインプットします．そのアドレスの内容が表示され，入力待ちになります．内容を変更するときは，1～2桁の16進数をインプットします．変更しないときは，[RETURN]キーのみを押すと，次のアドレス内容を表示してきます．

アドレスの指定で16進数コード以外のキーをインプットすると，モニタコマンドに戻ります．

Mコマンド使用例

Mコマンドでダンプ	39，3Aを変更	変更後ダンプ
*M38	*M38	*M38
0038 CB-	0038 CB-	0038 CB-
0039 00-	0039 00-12	0039 12-
003A 00-	003A 00-34	003A 34-
003B 71-	003B 71-	003B 71-
003C CB-	003C CB-	003C CB-
003D FF-	003D FF-	003D FF-

このような操作によって，機械語のプログラムを作成することができます．

◆ G

指定のアドレスプログラムの実行を移します．

G［分岐アドレス］

分岐アドレス1～4桁の16進数でインプットします．インプットされた数値をアドレスとして，プログラムを実行します．

◆ R

MPUのレジスタの内容を表示します．

R

MPUのレジスタの内容を表示し，指定により変更もできます．変更方法はMコマンドと同じです．

```
*R
CC  84-
A   00-
B   44-
DP  00-
X   AA8E-
Y   8B0C-
U   033A-
PC  AA93-

*
```

◆ D

指定アドレスより64バイト分の内容を表示します．

D［アドレス］

アドレスは1～4桁の16進数でインプットします．指定アドレスより64バイト分の内容を表示します．

アドレスの指定を省略した場合は，直前に表示されたアドレスの次のアドレスから，64バイト分の内容を表示します．

```
*D0000

0000 00 00 00 00 00 00 00 00
0008 00 00 00 00 00 00 00 00
0010 22 22 00 00 00 00 04 00
0018 00 00 00 00 04 72 04 6F
0020 B9 20 00 00 00 00 00 00
0028 00 00 00 00 00 0C 07 0C
0030 09 0C 09 0C 09 70 9F 71
0038 CB 00 00 71 CB FF FF 00
*
```

付　録

- キャラクタコード表
- F-BASIC のエラーメッセージ
- 非漢字一覧表
- JIS 第 1 水準漢字一覧表

● キャラクタコード表 ●

下位＼上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		DE	(space)	0	@	P	`	p				ー	タ	ミ		
1	SH	D1	!	1	A	Q	a	q			。	ア	チ	ム		円
2	SX	D2	"	2	B	R	b	r			「	イ	ツ	メ		年
3	EX	D3	#	3	C	S	c	s			」	ウ	テ	モ		月
4	ET	D4	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	EQ	NK	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	AK	SN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	BL	EB	▼	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	BS	CN	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	〒
9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	市
A	LF	SB	*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	区
B	HM	EC	+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	町
C	CL	→	,	<	L	¥	l	|			ャ	シ	フ	ワ	●	村
D	CR	←	-	=	M	]	m	}			ュ	ス	ヘ	ン	○	人
E	SO	↑	.	>	N	^	n	‾			ョ	セ	ホ	゛		
F	SI	↓	/	?	O	_	o	DL			ッ	ソ	マ	゜		

● F-BASICのエラーメッセージ ●

エラーコード	エラーメッセージ	内　　容
01	Next Without For	NEXTに対応するFOR文がない．
02	Syntax Error	コマンドまたは，文の書き方に誤まりがある．
03	Return Without Gosub	GOSUB文によって呼出されていないのに，RETURN文に出会った．
04	Out Of Data	READ文によって読込むべきデータがない．
05	Illegal Function Call	関数やステートメントの呼び方に誤まりがある．
06	Overflow	整数値または実数値が，許される範囲をこえている． または代入される数値が大きすぎる． 整数値のとき　−32768〜32767　の範囲にない． 実数値のとき　−1.70141E+38〜1.70141E+38　の範囲にない．
07	Out Of Memory	メモリが足りなくなった．
08	Undefined Line Number	指定された行番号が定義されていない．
09	Subscript Out Of Range	配列の添字が0から上限の範囲にない．
10	Duplicate Definition	同じ名前の配列または，ユーザ関数を2度宣言している．
11	Division By Zero	除算の分母が0である．
12	Illegal Direct	直接モードで使えないステートメントを用いた．
13	Type Mismatch	変数または定数の型が合わない．文字と数値を演算しようとしている． 代入の左辺と右辺，関数の引数の型，……
14	Out Of String Space	文字領域が足りなくなった．
15	String Too Long	文字定数が256文字をこえている．または文字式の結果が256文字以上になった．
16	String Formula Too Complex	文字式が複雑すぎる．
17	Can't Continue	CONTコマンドによるプログラムの続行ができない．
18	Undefined User Function	定義されていない関数を参照している．
19	NO Resume	エラー処理ルーチンにRESUMEがない．
20	Resume without Error	エラーが起きていないのに，RESUME文を実行しようとした．
21	Unprintable Error	エラーメッセージの定義されていないエラーを出そうとした．
22	Missing Operand	必要なオペランドが抜けている．
23	For without Next	FOR〜NEXTの対応が正しくない．
24	While without Wend	WHILE文に対応するWEND文がない．
25	Wend without While	WEND文に対応するWHILE文がない．
26	Bubble Full	バブルカセットがいっぱいであり，データの登録ができない．
50	Bad File Number	ファイル番号が誤っている．
51	Bad File Mode	入力モードでオープンしたファイル番号に対して出力しようとした．または，出力モードでオープンしたファイル番号から入力しようとした．
52	File Already Open	ファイルを二重にオープンしようとした．

エラーコード	エラーメッセージ	内　　容
53	Device I/O Error	使用したデバイスに入出力エラーが発生した.
54	Input Past End	ファイルの全てのデータを読んだ後に, INPUT 文を実行した.
55	Bad File Descriptor	ファイルディスクリプタの記述に誤りがある.
56	Direct Statement In File	アスキー形式のプログラムファイル中に, 直接ステートメントがあった.
57	File Not Open	ファイルがオープンされていない.
58	Bad Data In File	ファイル上のデータの形式が間違っている.
59	Device In Use	使用中のデバイスに対して, 再度オープンしようとした.
60	Device Unavailable	I/O デバイスが入出力可能な状態にない.
61	Buffer Overflow	入出力バッファがオーバーフローした.
62	Protected Program	保護されているプログラムに ,書込み修正を行おうとした.
63	File Not Found	指定されたファイル名が見つからない.
64	File Already Exists	指定されたファイル名はすでに存在している.
65	Directory Full	ディレクトリ領域がいっぱいであり, 新たなファイルの登録ができない.
66	Too Many Open Disk Files	確保されているファイルの個数をこえてオープンしようとした.
67	Disk Full	ディスクがいっぱいであり, データの登録ができない.
68	Field Overflow	フィールドの長さが 256 バイトをこえている.
69	String Not Fielded	Field 文で宣言された文字変数以外の変数に LSET, RSET を用いて代入しようとしている.
70	Bad Record Number	指定されたレコード番号は存在しない.
71	Bad File Structure	ファイルの構成に誤りがある.
72	Drive Not Ready	指定されたドライブ番号は Ready 状態にはない.
73	Disk Write Protected	Disk が書込み保護されている.

備考：エラーコード 65～73 は, Disk に関するエラーメッセージです.

● 非漢字一覧表 ●

コードは全て16進形式

記号

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
212X			、	。	，	．	・	：	；	？	！	゛	゜	´	｀	¨
213X	＾	￣	＿	ヽ	ヾ	ゝ	ゞ	〃	仝	々	〆	〇	ー	―	‐	／
214X	＼	～	‖	｜	…	‥	‘	’	“	”	（	）	〔	〕	［	］
215X	｛	｝	〈	〉	《	》	「	」	『	』	【	】	＋	－	±	×
216X	÷	＝	≠	＜	＞	≦	≧	∞	∴	♂	♀	°	′	″	℃	￥
217X	＄	￠	￡	％	＃	＆	＊	＠	§	☆	★	○	●	◎	◇	
222X		◆	□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	〓	

（例えば，％のコードは2173と読みます．実際の使用には“&H”をつけて，「&H2173」とします）

英・数字

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
233X	０	１	２	３	４	５	６	７	８	９						
234X		Ａ	Ｂ	Ｃ	Ｄ	Ｅ	Ｆ	Ｇ	Ｈ	Ｉ	Ｊ	Ｋ	Ｌ	Ｍ	Ｎ	Ｏ
235X	Ｐ	Ｑ	Ｒ	Ｓ	Ｔ	Ｕ	Ｖ	Ｗ	Ｘ	Ｙ	Ｚ					
236X		ａ	ｂ	ｃ	ｄ	ｅ	ｆ	ｇ	ｈ	ｉ	ｊ	ｋ	ｌ	ｍ	ｎ	ｏ
237X	ｐ	ｑ	ｒ	ｓ	ｔ	ｕ	ｖ	ｗ	ｘ	ｙ	ｚ					

ひらがな

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
242X		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
243X	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
244X	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
245X	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
246X	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
247X	ゐ	ゑ	を	ん												

カタカナ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
252X		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
253X	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
254X	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
255X	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
256X	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
257X	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									

ギリシャ文字

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
262X		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
263X	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
264X		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
265X	π	ρ	σ	τ	υ	φ	χ	ψ	ω							

ロシア文字

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
272X		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
273X	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
274X	Ю	Я														
275X		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
276X	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
277X	ю	я														

● JIS 第1水準漢字一覧表 ●

コードは全て 16 進形式

ア

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
302X		亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥
303X	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或
304X	粟	袷	安	庵	按	暗	案	闇	鞍	杏						

（例えば，安のコードは3042と読みます．実際の使用には "&H" をつけて，「&H 3042」とします）

イ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
304X											以	伊	位	依	偉	囲
305X	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃
306X	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
307X	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	
312X		院	陰	隠	韻	吋										

ウ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
312X							右	宇	烏	羽	迂	雨	卯	鵜	窺	丑
313X	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運
314X	雲															

エ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
314X		荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
315X	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円
316X	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁
317X	艶	苑	薗	遠	鉛	鴛	塩									

オ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
317X								於	汚	甥	凹	央	奥	往	応	
322X		押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億
323X	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音				

カ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
323X													下	化	仮	何

次頁につづく

カ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
324X	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
325X	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨
326X	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕
327X	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	
332X		魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾
333X	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙
334X	垣	柿	蠣	鈎	劃	嚇	各	廓	拡	攪	格	核	殻	獲	確	穫
335X	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫
336X	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹
337X	叶	椛	樺	鞄	株	兜	竈	蒲	釜	鎌	嚙	鴨	栢	茅	萱	
342X		粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦
343X	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓
344X	汗	漢	澗	灌	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
345X	莞	観	諌	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸
346X	巖	玩	癌	眼	岩	翫	贋	雁	頑	顔	願					

キ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
346X												企	伎	危	喜	器
347X	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	
352X		機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
353X	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戲	技	擬	欺	犠	疑
354X	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
355X	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救
356X	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居
357X	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	
362X		供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強
363X	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷
364X	鏡	響	饗	驚	仰	凝	堯	暁	業	局	曲	極	玉	桐	粁	僅
365X	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟
366X	謹	近	金	吟	銀											

ク

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
366X						九	倶	句	区	狗	玖	矩	苦	軀	駆	駈
367X	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	
372X		掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君
373X	薫	訓	群	軍	郡											

ケ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
373X						卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型
374X	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
375X	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頸	鶏	芸	迎	鯨
376X	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件
377X	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	
382X		検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣
383X	鍵	険	顕	験	鹼	元	原	厳	幻	弦	減	源	玄	現	絃	舷
384X	言	諺	限													

コ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
384X				乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
385X	湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互
386X	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐
387X	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	
392X		后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康
393X	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪
394X	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
395X	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	礦	鋼	閤	降
396X	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麴	克	刻
397X	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	
3A2X		此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕
3A3X	紺	艮	魂													

サ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3A3X				些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖
3A4X	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
3A5X	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在
3A6X	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削
3A7X	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	
3B2X		察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒
3B3X	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛
3B4X	酸	餐	斬	暫	残											

シ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3B4X						仕	仔	伺	使	刺	司	史	嗣	四	士	始
3B5X	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止
3B6X	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌
3B7X	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	
3C2X		次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿
3C3X	式	識	鴫	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆
3C4X	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
3C5X	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵
3C6X	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種
3C7X	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	
3D2X		宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐
3D3X	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎
3D4X	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
3D5X	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳
3D6X	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒
3D7X	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	
3E2X		勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少
3E3X	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢
3E4X	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
3E5X	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤

次頁につづく

シ	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3E6X	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常
3E7X	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	
3F2X		拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵
3F3X	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真
3F4X	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
3F5X	塵	壬	尋	甚	尽	腎	訊	迅	陣	靭						

ス	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3F5X											笥	諏	須	酢	図	厨
3F6X	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随
3F7X	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	
402X		澄	摺	寸												

セ	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
402X					世	瀬	畝	是	凄	制	勢	姓	征	性	成	政
403X	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠
404X	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
405X	石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設
406X	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦
407X	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	
412X		繊	羨	腺	舛	船	薦	詮	賎	践	選	遷	銭	銑	閃	鮮
413X	前	善	漸	然	全	禅	繕	膳	糎							

ソ	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
413X										噌	塑	岨	措	曾	曽	楚
414X	狙	疏	疎	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
415X	双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻
416X	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡
417X	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	
422X		臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗

次頁につづく

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ソ	423X	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜		

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
タ	423X															他	多
	424X	太	汰	詑	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
	425X	対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸
	426X	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄
	427X	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	
	432X		叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰
	433X	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽
	434X	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談			

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
チ	434X														値	知	地
	435X	弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄
	436X	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷
	437X	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	
	442X		帖	帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺
	443X	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕
	444X	沈	珍	賃	鎮	陳											

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ツ	444X						津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
	445X	槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壷	嬬	紬	爪	吊
	446X	釣	鶴														

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
テ	446X			亭	低	停	偵	剃	貞	呈	堤	定	帝	底	庭	廷	弟
	447X	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	
	452X		邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲

次頁につづく

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
テ	453X	徹	撤	轍	迭	鉄	典	塡	天	展	店	添	纒	甜	貼	転	顚
	454X	点	伝	殿	澱	田	電										

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ト	454X							兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
	455X	登	菟	賭	途	都	鍍	砥	礪	努	度	土	奴	怒	倒	党	冬
	456X	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	檮	棟
	457X	盗	淘	湯	濤	灯	燈	当	痘	禱	等	答	筒	糖	統	到	
	462X		董	蕩	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働
	463X	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿
	464X	得	徳	瀆	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
	465X	鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ナ	466X	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	畷	南	楠
	467X	軟	難	汝													

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ニ	467X				二	尼	弐	邇	匂	賑	肉	虹	廿	日	乳	入	
	472X		如	尿	韮	任	妊	忍	認								

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ヌ	472X									濡							

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ネ	472X										禰	祢	寧	葱	猫	熱	年
	473X	念	捻	撚	燃	粘											

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ノ	473X						乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿
	474X	農	覗	蚤													

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ハ	474X				巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
	475X	俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅
	476X	楳	煤	狽	買	売	賠	陪	這	蝿	秤	矧	萩	伯	剥	博	拍
	477X	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	
	482X		函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発
	483X	醗	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反
	484X	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
	485X	釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮					

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ヒ	485X												匪	卑	否	妃	庇
	486X	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被
	487X	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	
	492X		鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼
	493X	桧	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹
	494X	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
	495X	賓	頻	敏	瓶												

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
フ	495X					不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷
	496X	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫
	497X	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	
	4A2X		福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳
	4A3X	憤	扮	焚	奮	粉	糞	紛	雰	文	聞						

ヘ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4A3X											丙	併	兵	塀	幣	平
4A4X	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
4A5X	偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭			

ホ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4A5X														保	舗	鋪
4A6X	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣
4A7X	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	
4B2X		法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒
4B3X	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某
4B4X	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	卜	墨
4B5X	撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆

マ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4B6X	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槙	幕	膜	枕	鮪	柾
4B7X	鱒	桝	亦	俣	又	抹	末	沫	迄	侭	繭	麿	万	慢	満	
4C2X		漫	蔓													

ミ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4C2X				味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙
4C3X	粍	民	眠													

ム

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4C3X				務	夢	無	牟	矛	霧	鵡	椋	婿	娘			

メ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4C3X														冥	名	命
4C4X	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺		

モ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4C4X															摸	模
4C5X	茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅
4C6X	尤	戻	籾	貰	問	悶	紋	門	匁							

ヤ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4C6X										也	冶	夜	爺	耶	野	弥
4C7X	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓					

ユ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4C7X												愉	愈	油	癒	
4D2X		諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧
4D3X	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕			

ヨ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4D3X														予	余	与
4D4X	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
4D5X	熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遥	陽	養	慾	抑	欲
4D6X	沃	浴	翌	翼	淀											

ラ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4D6X						羅	螺	裸	来	莱	頼	雷	洛	絡	落	酪
4D7X	乱	卵	嵐	欄	濫	藍	蘭	覧								

リ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4D7X									利	吏	履	李	梨	理	璃	
4E2X		痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜
4E3X	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌
4E4X	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
4E5X	緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟				

ル		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	4E5X													瑠	塁	涙	累
	4E6X	類															

レ		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	4E6X		令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗
	4E7X	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	
	4F2X		蓮	連	錬												

ロ		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	4F2X					呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗
	4F3X	楼	榔	浪	漏	牢	狼	篭	老	聾	蝋	郎	六	麓	禄	肋	録
	4F4X	論															

ワ		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	4F4X		倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨
	4F5X	椀	湾	碗	腕												

索引